Python Package for World Wide Statistics Visualization

Name 1, Name 2, ...

Université de Montpellier

April 24, 2021

Outline

Outline

Introduction

Motivation of this work

Introduction

Quick Example

```
from wwstatviz import Visualizer

v = Visualizer('/path/to/data.csv')
fig = v.choropleth(title = '...',
                   feature = 'GDP',
                   countries = ['FRA', 'USA', 'AFG', ...])
fig.save('/path/to/output_file.png')
```

```
fig = v.heatmap(title = '...',
                features = 'all',
                countries = ['ALG', 'GER', 'SEN', ...])
fig.show() # for inline display (in browsers for example)
```

Outline

API

Available plots in the current state are:

- choropleth: ...
- heatmap: ...
- line plot: ... for time series visualization
- histogram: ...

Package structure

```
wwstatviz
|-- __init__.py
|-- generators/
|   |-- __init__.py
|   |-- choropleth.py
|   |-- generator.py
|   |-- heatmap.py
|   `-- line.py
|-- io/
|   |-- __init__.py
|   |-- csvreader.py
|   |-- jsonreader.py
|   |-- iso.py
|   |-- reader.py
|   `-- writer.py
|-- figure.py
`-- visualizer.py
```

The Visualizer Class

Input Data

The constructor of the class takes as input a data file (in the CSV format):

- The first line must contain the header
- Each row must start with a country code (ISO-3166 2-digit or 3-digit)
- The columns represent the features of the data

Example:

```
,f1,f2,f3
AFG,0,1,2
BEL,5,4,3
FRA,6,7,8
SEN,12,13,14
USA,3,33,8
```

The Visualizer Class

Main functions

- The Visualizer class is the main interface of the API (it orchastrates the differents tasks/actions).
- The constructor (__init__(self, data_path)) takes as input the path to the data file, and calls the corresponding reader from the “io” subpackage
- It contains functions (methods) for generating graphics (choropleth, histogram, etc.)
- These function are simple calls to the Generators
- Each function returns a Figure object (for later use, show()/save())

Generators

About Generators:

- Generators are responsible for producing plots.
- Each generator should inherit from the base class “Generator” and must implement a “generate()” method
- The generator constructor takes as argument the different options to be used for generating the plots (e.g. whether or not to draw a legend, the countries to use, etc.)

Example:

```
1   class XYZGenerator(Generator): # class inheritance
2
3       def __init__(self, ...): # this is the contructor
4           ...
5
6       # generate method must be implemented
7       # and must return a figure
8       def generate(self):
9           ...
10          return figure
```

Input/Output module

About the “io” module:

- The “io” module is responsible for:
  - reading data files from disk for different formats (csv, json, etc.)
  - writing generatred figures to disk
- It contains the base classes Reader and Writer
- For each data format, a reader submodule must be implemented
- Each reader submodule (e.g. csvreader) should implement a class that:
  - inherits from the base class “Reader”
  - implements a “read()” method

Outline

A Web Application

About Flask

Outline

Unit Testing

- Testing technique
- Runs on individual units or components (functions, methods, etc.)
- Validate correctness of code
- Verify if code performs as expected

Unit Tests

Unit Testing Using pytest

The tests are performed through assertions:

- Whether or not the figure is generated
- The instance of the generated plot (a matplotlib figure, a plotly figure, etc.)
- The writing of the generated figure in disk

Example:

```
    v = Visualizer('/workspace/data/test_cc_3d.csv')
    fig = v.heatmap(title = 'This is a test heatmap',
                    xlabel = 'Countries', ylabel = 'Countries')
    assert fig.figure is not None
    assert isinstance(fig.figure, matplotlib.figure.Figure)
    fig.save('test_heatmap.png')
    assert Path('test_heatmap.png').is_file()
```

Figure: Unit Test Exection

Outline

Choropleth

Heatmap

Time Series Plot

Histogram

Outline