

Python Package for World Wide Statistics Visualization

Sophie Manuel, Ravahere Paint-Koui, Seydou Sane, Anas Zakroum

Université de Montpellier

April 25, 2021



Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Introduction

Motivation of this work

Many python packages for generating plots (matplotlib, plotly, etc.):

- ▶ Have lot of information to remember
- ▶ Have very long documentation
- ▶ Take time to get used to

What about aggregating all of these packages into one package that is :

- ▶ Task oriented
- ▶ Easy to use

Introduction

Quick Example

```
1  from wwstatviz import Visualizer
2
3  v = Visualizer('/path/to/data.csv')
4  fig = v.choropleth(title = '...',
5                    feature = 'GDP',
6                    countries = ['FRA', 'USA', 'AFG', ...])
7  fig.save('/path/to/output_file.png')
```

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

Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Available plots in the current state are:

- ▶ choropleth: provides an easy way to visualize how an indicator varies accross a region;
- ▶ heatmap: used to visualize how correlated are variables;
- ▶ line plot: represents the time evolution of an indicator;
- ▶ histogram: plots the values of given indicators by country.

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 orchestrates the different tasks/actions).
- ▶ The constructor (`__init__(self, data_path)`) takes as inputs 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 functions 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 constructor
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 generated 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

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Continuous Integration

- ▶ Automating integration of code changes
- ▶ From multiple collaborators
- ▶ Into a single project
- ▶ Avoid merge problems

Continuous Integration

wwstatviz

- ▶ Check requirements and install dependencies
- ▶ Test with pytest
- ▶ Check for PEP8

Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Unit Testing

- ▶ **F**ast
- ▶ **I**solated
- ▶ **R**epetable
- ▶ **S**elf-validating
- ▶ **T**imely

Unit Tests

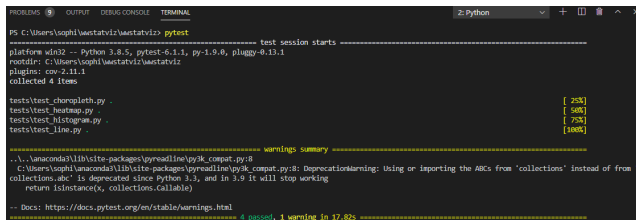
Unit Testing Using pytest

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:

```
1 v = Visualizer('/workspace/data/test_cc_3d.csv')
2 fig = v.heatmap(title = 'This_is_a_test_heatmap',
3                 xlabel = 'Countries', ylabel = 'Countries')
4 assert fig.figure is not None
5 assert isinstance(fig.figure, matplotlib.figure.Figure)
6 fig.save('test_heatmap.png')
7 assert Path('test_heatmap.png').is_file()
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
z: Python
PS C:\Users\sophi\workspace\workspace> pytest
===== test session starts =====
platform win32 -- Python 3.8.5, pytest-6.1.1, py-1.9.0, pluggy-0.13.1
rootdir: C:\Users\sophi\workspace\workspace
plugins: cov-2.11.1
collected 4 items

tests\test_choropleth.py . [ 25%]
tests\test_heatmap.py . [ 50%]
tests\test_histogram.py . [ 75%]
tests\test_line.py . [100%]

===== warnings summary =====
..\..\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8
C:\Users\sophi\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated since Python 3.3, and in 3.9 it will stop working
  return isinstance(x, collections.Callable)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== 4 passed, 1 warning in 17.82s =====
```

Figure: Unit Test Execution

Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Choropleth

Generates a choropleth map of given list of countries and indicator.'


```
1     v = Visualizer('path/to/file.csv')
2     fig = v.choropleth(title = '...',
3                       features = 'desired_feature',
4                       countries = 'all')
5     fig.show() # for inline display (in browsers for example)
```

beamer/inc/graphics/choropleth.png

Heatmap

Generates a heatmap correlation matrix and allows to show in a glance which countries are correlated.

```
1  v = Visualizer('path/to/file.csv')
2  fig = v.heatmap(countries='all', features='all',
3                  method='pearson', mask=True,
4                  title='This is a test heatmap', xlabel='
                    Countries', ylabel='Countries')
5  fig.show() #for inline display (in browsers for example)
```



beamer/inc/graphics/heatmap.png

Time Series Plot

Generates a lineplot of given lists of countries and indicators.

```
1  v = Visualizer('path/to/file.csv')
2  fig = v.line(countries='all', features='all',
3              title='This is a test of line plot', xlabel='Label
4              of x axis', ylabel='Label of y axis',
5              legend=True)
6  fig.show() # for inline display (in browsers for example)
```

beamer/inc/graphics/line.png

Histogram

Generates a histogram of given lists of countries and indicators.

```
1  v = Visualizer('path/to/file.csv')
2  fig = v.histogram(countries=['MAR', 'CHN', 'FRA', 'RUS', '
    PYF', 'SEN'],
3                      features=['f1', 'f2', 'f3'],
4                      title='This is a test histogram', xlabel='
    Label of x axis', ylabel='Label of y axis
    ',
5                      legend=True)
6  fig.show() # for inline display (in browsers for example)
```

beamer/inc/graphics/histogram.png

Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

A Web Application

The wwstatviz Web Application

We wanted to develop a Web Application that would :

- ▶ be coded using the Python package Flask;
- ▶ facilitate the displaying of the figures;
- ▶ save the user from writing any Python code.

Current display options

Currently, the figures are showed:

- ▶ on a web page : choropleth map;
- ▶ in a Python Shell : heatmap, lineplot and histogram.

Outline

Introduction

Package wwstatviz - API

Continuous Integration

Unit Tests

Demo

Package wwstatviz-webapp - User Interface

Conclusion

Conclusion

What we have learned

- ▶ Learn more about geospatial visualization techniques;
- ▶ Create one package from several Python packages;
- ▶ Manipulate Python classes and link them.

What can be improved

- ▶ End the development of the web application;
- ▶ More custom options for the figures we are able to create.

Thank you for your attention!

You can get more information about wwstatviz on:

► [wwstatviz github](#)