

# Python Package for World Wide Statistics Visualization

Name 1, Name 2, ...

Université de Montpellier

April 23, 2021

# Outline

Introduction

Package wwstatviz - API

Package wwstatviz-webapp - User Interface

Demo

Conclusion

# Outline

Introduction

Package wwstatviz - API

Package wwstatviz-webapp - User Interface

Demo

Conclusion

# Introduction

Motivation of this work

# 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

Package wwstatviz-webapp - User Interface

Demo

Conclusion

Available plots in the current state:

- ▶ 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 take 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 (i.e. the features) represent 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

- ▶ This Visualizer class is the main interface of the API.
- ▶ The constructor (`__init__(self, data_path)`) takes as input the path to the data file
- ▶ It contains functions for generating graphics (choropleth, histogram, etc.)
- ▶ These function are simple calls to the Generators
- ▶ Each function returns a Figure object (for later use)

# 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

Package wwstatviz-webapp - User Interface

Demo

Conclusion

# Outline

Introduction

Package wwstatviz - API

Package wwstatviz-webapp - User Interface

Demo

Conclusion

# Outline

Introduction

Package wwstatviz - API

Package wwstatviz-webapp - User Interface

Demo

Conclusion