pyradon Documentation

Release master

Falagas Alekos

Oct 29, 2020

Contents

CHAPTER 1

About

A web based development enviroment for scientists to design, develop and test the correlation between radon measurements and earthquake data. The language that is used is Python, one of the most popular languages in data science. Jupyter Hub is a multi-user development environment using the IPython interface, allowing code implementation and interpretation using a web interface.

CHAPTER 2

Endpoint

The service is available at atmo-seism.neanias.eu

CHAPTER 3

Access

The service is using NEANIAS Identity Provider for authentication.

CHAPTER 4

Usage

After successful login, Jupyter is automatically launched. Python 3 programming language is supported (version 3.6.9). The following libraries are preinstalled:

- numpy
- scipy
- matplotlib
- sklearn
- scikit-image
- scikit-learn
- pandas
- cartopy
- shapely
- pyradon (module created for the purposes of the project. For more information see bellow)

CHAPTER 5

Pyradon Package

5.1 pyradon's Documentation

class pyradon.**Pyradon**

A collection of methods for the corellation of soil radon measurements as a potential tracer of tectonic and volcanic activity.

static check_coords(*lat*, *lon*, *file*)

This method is checking the validity of the coordinates.

Parameters

- **lat** (*Series*) – Latitude data in pandas series
- **lon** (*Series*) – Longitude data in pandas series
- **file** (*str*) – Path-Pathlike string that indicates the file

Raises

- **ValueError** – If latitude is not between -90 and 90 degrees
- **ValueError** – If longitude is not between -180 and 180 degrees

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> # Path to file
>>> #gas_file = "./Data/01_2016_2017_PP_Radon_Etna.csv"
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading Gas Data
>>> gas_data = pd.read_csv(gas_file)
>>> # Call check coordinates for gas data
>>> Pyradon.check_coords(gas_data['Lat'], gas_data['Lon'], gas_file)
```

static clear_data(*df*)

The method that clears data from null values.

Parameters **df** (*pandas.DataFrame*) – The data in pandas dataframe format

Returns Returns a pandas dataframe with the data cleaned by NaN values

Type pandas.DataFrame

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> # Path to file
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading Gas Data
>>> gas_data = pd.read_csv(gas_file)
>>> # Call the method
>>> gas_data = Pyradon.clear_data(gas_data)
```

static disp_results(*pred*, *vars*, *coef*, *intercept*, *r2*)

A method to display the analysis results.

Parameters

- **pred** (*Series*) – Pandas Series with the dependent variable
- **vars** (*DataFrame*) – Pandas DataFrame with the independent variables
- **coef** (*ndarray*) – Coeffecients of the model.
- **intercept** (*Intercept of the model*) – ndarray
- **r2** (*float*) – R2 score of the model

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> # Path to file
>>> #gas_file = "./Data/01_2016_2017_PP_Radon_Etna.csv"
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading Gas Data
>>> gas_data = pd.read_csv(gas_file)
>>> # Spliting data to x and y values
>>> X = gas_data[['Temperature', 'Pressure']]
>>> Y = gas_data[['Radon']]
>>> Y_pred, intercept, coef, score = Pyradon.regression(X, Y)
>>> # Call display results function
>>> Pyradon.disp_results(Y.columns, X.columns, coef, intercept, score)
```

static normal_probplot(*data*, *x_label=None*, *y_label=None*, ***kwargs*)

Generates a probability plot of sample data against the quantiles of normal theoretical distribution.

Parameters

- **data** (*array-like*) – Sample data from which probplot creates the plot
- **x_label** (*str, optional*) – Label of x-axis, defaults to None
- **y_label** (*str, optional*) – Label of y-axis, defaults to None
- ****kwargs** – See bellow for a list of valid properties

Name	Description	Type
lc	Color of line	str
mc	Color of cicle	str
lw	Width of line	float
mw	Width of cicle	float
labelsize	Size of the labels	float
title	Title of the chart	str
titlesize	Size of the title	float
fontcolor	Color of the fonts	str

Based on matplotlib (matplotlib.org/)

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> from sklearn import linear_model
>>> # Path to file
>>> #gas_file = "./Data/01_2016_2017_PP_Radon_Etna.csv"
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading Gas Data
>>> gas_data = pd.read_csv(gas_file)
>>> # Spliting data to x and y values.
>>> X = gas_data[['Temperature', 'Pressure']]
>>> Y = gas_data[['Radon']]
>>> regr = linear_model.LinearRegression()
>>> # Create model
>>> regr.fit(X, Y)
>>> Y_pred = regr.predict(X)
>>> # Convert the predicted results back to pandas dataframe
>>> Y_pred = pd.DataFrame(Y_pred)
>>> Y_pred.columns = ['Predicted Radon']
>>> # Add Predicted values to the initial dataframe
>>> gas_data = pd.merge(gas_data, Y_pred, left_index = True, right_index=True)
>>> # Filtering Radon values
>>> gas_data['Filtered Radon'] = gas_data['Radon'] - gas_data['Predicted Radon
↪']
>>> Pyradon.normal_probplot(gas_data['Filtered Radon'], lc = 'g', lw = 3.,␣
↪labelsize = 28.)
>>> Pyradon.normal_probplot(gas_data['Filtered Radon'], mc = 'r', mw = 4., lc␣
↪= 'g', lw = 1.2, labelsize = 9., title = 'Probability Plot of Filtered Radon
↪', titlesize = 13.,        ...    fontcolor = 'r')
>>> Pyradon.normal_probplot(gas_data['Filtered Radon'], mc = 'b', mw = 9., lc␣
↪= 'g', lw = 2., labelsize = 9., title = 'Probability Plot of Filtered Radon
↪', titlesize = 13.)
```

static plot_different_sized_cicles(*x*, *y*, *grouped*, *values*, *labels*, *x_label=None*, *y_label=None*, ***kwargs*)

Plots different sized cicles of a grouped dataframe on a map.

Parameters

- **x** (*str*) – Column name of the DataFrame that contains the longitude values
- **y** (*str*) – Column name of the DataFrame that contains the latitude values
- **grouped** (*pandas.DataFrameGroupBy*) – Grouped dataset
- **values** (*list*) – Values that data are grouped and also the cicle size of each category

- **labels** (*list*) – Labels to be showed in the legend
- **x_label** (*str, optional*) – Label of x-axis, defaults to None
- **y_label** (*str, optional*) – Label of y-axis, defaults to None
- ****kwargs** – See bellow for a list of valid properties

Name	Description	Type
c	Color	str
alpha	The degree of transparency	str
legendsize	Size of the legend	float
labelsize	Size of the labels	float
title	Title of the chart	str
titlesize	Size of the title	float
fontcolor	Color of the fonts	str

Based on matplotlib (matplotlib.org/)

Example

```
>>> import pandas as pd
>>> import numpy as np
>>> from pyradon import Pyradon
>>> # Path to file
>>> earthquake_file = "./Data/02_Earthquakes_Etna_UTM.csv"
>>> #earthquake_file = "./Data/test_earth2.csv"
>>> # Reading Gas Data
>>> earthquake_data = pd.read_csv(earthquake_file)
>>> # Categorize data based on the earthquake magnitude
>>> conditions = [(earthquake_data['Magnitude'] >= 4.0), ((earthquake_data[
↪'Magnitude'] >= 3.0) & (earthquake_data['Magnitude'] < 4.0)), ((earthquake_
↪data['Magnitude'] >= 2.0) & (earthquake_data['Magnitude'] < 3.0)),␣
↪(earthquake_data['Magnitude'] < 2.0)]
>>> values = [50, 100, 200, 400]
>>> # Adding a new column with the classification result
>>> earthquake_data['Category'] = np.select(conditions, values)
>>> # Creating labels
>>> labels = ['<=1.9', '2.0 - 2.9', '3.0 - 3.9', '>= 4.0']
>>> # Creating a new dataframe with the data grouped by the Category column
>>> grouped = earthquake_data.groupby('Category')
>>> Pyradon.plot_different_sized_cicles('Lon', 'Lat', grouped, values, labels)
>>> Pyradon.plot_different_sized_cicles('Lon', 'Lat', grouped, values, labels,
↪ x_label = 'Longitude', y_label = 'Latitude', c = 'g', alpha = 0.4,␣
↪labelsize = 12., fontcolor = 'r')
>>> Pyradon.plot_different_sized_cicles('Lon', 'Lat', grouped, values, labels,
↪ x_label = 'Longitude', y_label = 'Latitude', c = 'g', alpha = 0.4,␣
↪labelsize = 12., fontcolor = 'r',          ...     title = 'Earthquake␣
↪Magnitude', titlesize = 14., legendtitle = 'Magnitude')
```

static plotline(*x*, *y*, *x_label=None*, *y_label=None*, ***kwargs*)

A matplotlib based function for plotting a simple line.

Parameters

- **x** (*Series*) – Data to be plotted on x axis
- **y** (*Series*) – Data to be plotted on y axis
- **x_label** (*str, optional*) – Label of x axis, defaults to None

- **y_label** (*str, optional*) – Label of y axis, defaults to None
- ****kwargs** – See bellow for a list of valid properties

Name	Description	Type
lc	Color of line	str
lw	Width of line	float
labelsize	Size of the labels	float
title	Title of the chart	str
titlesize	Size of the title	float
fontcolor	Color of the fonts	str

Based on matplotlib (matplotlib.org/)

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> from sklearn import linear_model
>>> # Path to file
>>> #gas_file = "./Data/01_2016_2017_PP_Radon_Etna.csv"
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading Gas Data
>>> gas_data = pd.read_csv(gas_file)
>>> # Spliting data to x and y values.
>>> X = gas_data[['Temperature', 'Pressure']]
>>> Y = gas_data[['Radon']]
>>> regr = linear_model.LinearRegression()
>>> # Create model
>>> regr.fit(X, Y)
>>> Y_pred = regr.predict(X)
>>> # Convert the predicted results back to pandas dataframe
>>> Y_pred = pd.DataFrame(Y_pred)
>>> Y_pred.columns = ['Predicted Radon']
>>> # Add Predicted values to the initial dataframe
>>> gas_data = pd.merge(gas_data, Y_pred, left_index = True, right_index=True)
>>> # Filtering Radon values
>>> gas_data['Filtered Radon'] = gas_data['Radon'] - gas_data['Predicted Radon
↪']
>>> x = pd.to_datetime(gas_data['Date / Hour'], format = '%m/%d/%Y %H:%M')
>>> y = gas_data[['Filtered Radon']]
>>> Pyradon.plotline(x, y)
>>> Pyradon.plotline(x, y, x_label = 'Date', y_label = 'Filtered Radon', lc =
↪'g', lw = 0.1, labelsize = 7.,        ...    title = 'Filtered Radon Time␣
↪Series', titlesize = 13., fontcolor = 'black')
```

static plotlinebar(*df1, df2, x_label=None, y_label=[None, None], **kwargs*)

A matplotlib based function that plots two lines on with different y axis and the same x axis in the same chart.

Parameters

- **df1** (*DataFrame*) – Contains data to be plotted on for the bars. X data must be on column 0 and Y data must be on column 1
- **df2** (*DataFrame*) – Contains data to be plotted on for the line. X data must be on column 0 and Y data must be on column 1

- **x_label** (*str, optional*) – Label of x-axis. The default name is the column name of the first column of df1 variable, defaults to None
- **y_label** (*list, optional*) – Label of y-axis for the two lines.The default name is the column name, defaults to [None, None]
- ****kwargs** – See bellow for a list of valid properties

Name	Description	Type
lc	Color of line	str
bc	Color of bar	str
lw	Width of line	float
bw	Width of bar	float
labelsize	Size of the labels	float
title	Title of the chart	str
titlesize	Size of the title	float
fontcolor	Color of the fonts	str

Based on matplotlib (matplotlib.org/)

Raises `ValueError` – Raises error when a y axis label is not provided for both dataframes

Example

```
>>> import pandas as pd
>>> import numpy as np
>>> from pyradon import Pyradon
>>> # Path to file
>>> earthquake_file = "./Data/02_Earthquakes_Etna_UTM.csv"
>>> #earthquake_file = "./Data/test_earth2.csv"
>>> # Reading Gas Data
>>> earthquake_data = pd.read_csv(earthquake_file)
>>> earthquake_data['Energy'] = earthquake_data['Magnitude'].apply(lambda x:␣
↪np.sqrt(10 ** (4.8 + 1.5 * x))) # Where energy in Joules^1/2
>>> # Sorting values based on date field after converting the Datetime column␣
↪to datetime object
>>> earthquake_data['Date'] = pd.to_datetime(earthquake_data.DateTime, format␣
↪= '%m/%d/%Y %H:%M')
>>> earthquake_data = earthquake_data.sort_values(by='Date')
>>> earthquake_data['Cumsum'] = earthquake_data.Energy.cumsum()
>>> # Creating a Date_only field to calculate the frequency per day
>>> earthquake_data['Date_only'] = earthquake_data.Date.dt.date
>>> counts = earthquake_data.Date_only.value_counts()
>>> counts = counts.to_frame()
>>> temp_df = earthquake_data.loc[earthquake_data.groupby('Date_only')['Cumsum
↪'].idxmax()]
>>> temp_df = temp_df.set_index('Date_only')
>>> temp_df = temp_df.merge(counts, right_index = True, left_index = True)
>>> temp_df = temp_df.rename(columns = {'Date_only': 'Number of events'})
>>> temp_df['Index'] = temp_df.index
>>> df1 = temp_df[['Index', 'Number of events']]
>>> df2 = temp_df[['Index', 'Cumsum']]
>>> Pyradon.plotlinebar(df1, df2, x_label = r'$Date$', y_label = [r'$Number\␣
↪of\ Events$', r'$J^{1/2}$'],  lc = 'g', lw = 2., bc = 'orange', bw = 4.,␣
↪fontcolor = 'red', title = 'Test', labelsize = 9.)
>>> del (temp_df, df1, df2)
```

static plotlines(*df1, df2, x_label=None, y_label=[None, None], **kwargs*)

A matplotlib based function that plots two lines on with different y axis and the same x axis in the same chart.

Parameters

- **df1** (*DataFrame*) – Contains data to be plotted on for the first line. X data must be on column 0 and Y data must be on column 1
- **df2** (*DataFrame*) – Contains data to be plotted on for the second line. X data must be on column 0 and Y data must be on column 1
- **x_label** (*str, optional*) – Label of x-axis. The default name is the column name of the first column of df1 variable, defaults to None
- **y_label** (*list, optional*) – Label of y-axis for the two lines. The default name is the column name for both, defaults to [None, None]
- ****kwargs** – See bellow for a list of valid properties

Name	Description	Type
l1c	Color of line 1	str
l2c	Color of line 2	str
l1w	Width of line 1	float
l2w	Width of line 2	float
labelsize	Size of the labels	float
title	Title of the chart	str
titlesize	Size of the title	float
fontcolor	Color of the fonts	str

Based on matplotlib (matplotlib.org/)

Raises **ValueError** – Raises error when a y axis label is not provided for both dataframes

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> # Path to file
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading data
>>> gas_data = pd.read_csv(gas_file)
>>> gas_data['Date / Hour'] = pd.to_datetime(gas_data['Date / Hour'], format␣
↪= '%m/%d/%Y %H:%M')
>>> df1 = gas_data[['Date / Hour', 'Temperature']]
>>> df2 = gas_data[['Date / Hour', 'Pressure']]
>>> # Plot Temperature and Pressure
>>> Pyradon.plotlines(df1, df2)
>>> Pyradon.plotlines(df1, df2, x_label = 'Date', y_label = [r'$Temperature\␣
↪(C^{o})$', None], l1c = 'g', l1w = 2.)
>>> Pyradon.plotlines(df1, df2, x_label = r'$Date$', y_label = [r'
↪$Temperature\ (C^{o})$', r'$Pressure\ (mBar)$'], l1c = 'g', l1w = 2., l2c =
↪'orange', l2w = 4.)
>>> Pyradon.plotlines(df1, df2, x_label = r'$Date$', y_label = [r'
↪$Temperature\ (C^{o})$', r'$Pressure\ (mBar)$'], l1c = 'g', l1w = 2., l2c =
↪'orange', l2w = 4., fontcolor = 'red', title = 'Test', labelsize = 9.)
```

static regression(*X*, *Y*)

Fits a linear model with coefficients w = (w1, . . . , wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters

- **X** (*DataFrame*) – Training data (independent variables)
- **Y** (*DataFrame, array-like*) – DataFrame, array-like

Returns A tuple with the predicted values, intercept value, coefficients and the R2 score of the model

Type tuple

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> # Path to file
>>> gas_file = "./Data/test_radon2.csv"
>>> # Reading Gas Data
>>> gas_data = pd.read_csv(gas_file)
>>> # Spliting data to x and y values.
>>> X = gas_data[['Temperature', 'Pressure']]
>>> Y = gas_data[['Radon']]
>>> Y_pred, intercept, coef, score = Pyradon.regression(X, Y)
```

static scatterplot (*x*, *y*, *x_label=None*, *y_label=None*, *invert_y_axis=True*, ***kwargs*)

A simple scatterplot of two variables with an invert y axis option.

Parameters

- **x** (*Series*) – Data to be plotted on x axis
- **y** (*Series*) – Data to be plotted on y axis
- **x_label** (*str, optional*) – Label of x axis, defaults to None
- **y_label** (*str, optional*) – Label of y axis, defaults to None
- **invert_y_axis** (*bool, optional*) – Invert y axis, defaults to True
- ****kwargs** – See bellow for a list of valid properties

Name	Description	Type
c	Color	str
alpha	The degree of transparency	str
s	Size of the object	float
legendsize	Size of the legend	float
labelsize	Size of the labels	float
title	Title of the chart	str
titlesize	Size of the title	float
fontcolor	Color of the fonts	str

Based on matplotlib (matplotlib.org/)

Example

```
>>> import pandas as pd
>>> from pyradon import Pyradon
>>> # Path to file
>>> earthquake_file = "./Data/02_Earthquakes_Etna_UTM.csv"
>>> #earthquake_file = "./Data/test_earth2.csv"
>>> # Reading earthquake data
```

(continues on next page)

(continued from previous page)

```
>>> earthquake_data = pd.read_csv(earthquake_file)
>>> Pyradon.scatterplot(earthquake_data['Lon'], earthquake_data['Depth'])
>>> Pyradon.scatterplot(earthquake_data['Lon'], earthquake_data['Depth'], x_
↪label = r'$Longitude\ ^{o}$', y_label = r'$Depth\ (km)$', invert_y_axis =␣
↪True,          ...     s = 20., c = 'g', alpha = 0.4, labelsize = 9.,␣
↪fontcolor = 'black',         ...     title = r'$Scatterplot\ with\ Depth\␣
↪and\ Longitude$', titlesize = 13.)
```

CHAPTER 6

References

- Neri, M., Ferrera, E., Giammanco, S. et al. Soil radon measurements as a potential tracer of tectonic and volcanic activity. Sci Rep 6, 24581 (2016). https://doi.org/10.1038/srep24581

Python Module Index

p

Index