pytsviz

Release 0.1.0

xtream

Oct 28, 2021

CONTENTS

CHAPTER

ONE

TABLE OF CONTENT

1.1 pytsviz

pytsviz is a suite of tools to quickly analyze and visualize time series data. It is partially based on the *tsviz* R package.

The *utils* module contains a set of useful utilities, not strictly related to visualization, we often use (e.g. harmonics computation).

The *viz* module contains functions for plotting univariate time series, as well as performing quick qualitative analyses such as decompositions, correlations and so on.

Some visualizations mimic the R packages *ggplot2* and *forecast*, as presented in the textbook *Forecasting: principles and practice* by Rob J. Hyndman and George Athanasopoulos. The online version of the text can be found here.

1.1.1 Install

The preferred way to install the package is using pip, but you can also download the code and install from source

To install the package using pip:

```
pip install pytsviz
```

1.1.2 Develop

After cloning, you need to install and setup Poetry. See instructions.

Then, inside the project directory, run:

```
poetry install
pre-commit install
```

Then, you're good to go.

You're free to submit your pull requests. Just make sure they follow conventional commit rules. This can be enforced by the *commitizen* tool, which is also included among the package dependencies.

Please also make sure that function documentation is consistent. We are currently using Sphinx docstrings.

1.1.3 Who we are

1.2 pytsviz package

pytsviz is a Python tool for quick visualization and analysis of time series. It is partially based on the tsviz R package. Is is still under heavy development, so feel free to point out any issues.

pytsviz.**plot_acf**(*df*, *y_col=None*, *time_col=None*, *partial=False*, *n_lags=None*, *title=None*, *show_threshold=True*, *show=True*, ***kwargs*)

Interactive bar plot of the auto-correlation function of a time series up to a certain lag.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **partial** (bool) – Whether to use the partial auto-correlation function.
- **n_lags** (Optional[int]) – Maximum number of time steps to consider as lag.
- **title** (Optional[str]) – Plot title.
- **show_threshold** (bool) – Whether to compute the ACF significance threshold, defined as the 95% confidence upper bound of a time series which has no auto-correlation, and show it as a horizontal line in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_decomposed_ts**(*df*, *method*, *y_col=None*, *time_col=None*, *title=None*, *subplots=True*, *show=True*, ***decomp_kwargs*)

Interactive plot of a decomposition of a time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **method** (Literal['STL', 'seasonal_additive', 'seasonal_multiplicative']) – Method of decomposition. It can be one of "STL", "seasonal_additive" or "seasonal_multiplicative".
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **subplots** (bool) – Whether to split the decomposition components into subplots.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **decomp_kwargs** (Any) – Keyword arguments for the selected decomposition method. See method for details.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_distribution**(*df*, *y_col=None*, *time_col=None*, *bins=None*, *title=None*, *show=True*)

Interactive histogram of a time series. Useful to assess marginal distribution shape.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **bins** (Optional[int]) – Number of bins in the histogram.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_extended_scatter_matrix**(*df*, *time_col=None*, *y_cols=None*, *title=None*, *show=True*)

Composite plot containing scatter matrix, distributions and couple-correlations of a set of time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_cols** (Optional[List[str]]) – Name of the columns from df to be used as variables in the plot. If None, all columns are used.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_forecast**(*df*, *y_col*, *y_hat_cols*, *lower_col=None*, *upper_col=None*, *time_col=None*, *title=None*, *show=True*)

Interactive plot of a time series forecast (or multiple forecasts).

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (str) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **y_hat_cols** (List[str]) – Name of the columns from df to be used as forecasts for the time series in the plot.

- **lower_col** (Optional[str]) – If given, name of the column to be used as lower confidence bound for the forecast.
- **upper_col** (Optional[str]) – If given, name of the column to be used as upper confidence bound for the forecast.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_gof**(*df*, *y_col*, *y_hat_col*, *time_col=None*, *acf_n_lags=None*, *pacf_n_lags=None*, *alpha=0.1*, *title='Goodness of Fit'*, *subplot_titles=('Actual vs Predicted Series', 'Actual vs Predicted Scatter', 'Residuals', 'Residuals ACF', 'Residuals PACF')*, *show=True*)

Interactive plot of goodness of fit visualizations. In order: Actual Series vs Predicted Series, Residuals series, and Actual vs Predicted scatter plot.

Parameters

- **acf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for ACF.
- **pacf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for PACF.
- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (str) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **y_hat_col** (str) – Name of the column from df to be used as forecast for the time series in the plot.
- **alpha** (float) – If a number is given, the confidence intervals for the given level are displayed in the plot.
- **title** (str) – Plot title.
- **subplot_titles** (Tuple[str, str, str, str, str]) – Tuple of titles for each of the 3 subplots.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Tuple of plotly figures if show is True, else nothing.

pytsviz.**plot_inverse_arma_roots**(*process*, *show=True*)

Plot of the inverse roots of a given ARMA process.

Parameters

- **process** (ArmaProcess) – The ARMA process (statsmodels.tsa.ArmaProcess) for which the roots are computed.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is `True`, else nothing.

pytsviz.**plot_psd**(*df*, *y_col=None*, *time_col=None*, *nfft=None*, *fs=1*, *min_period=0*, *max_period=inf*, *plot_time=False*, *title=None*, *show=True*, ***kwargs*)

Interactive histogram of the spectral density of a time series.

Parameters

- **df** (`DataFrame`) – Dataframe to use as the data source for the plot.
- **y_col** (`Optional[str]`) – Name of the column from df to be used as time series values in the plot.
- **time_col** (`Optional[str]`) – Name of the column from df to be used as time variable in the plot.
- **show** (`bool`) – Whether to call `Figure.show` at the end of the function. If `False`, returns the figure.
- **nfft** (`Optional[int]`) – Length of the FFT used. If `None` the length of df will be used.
- **fs** (`int`) – Sampling frequency of df.
- **min_period** (`int`) – Minimum period to consider.
- **max_period** (`int`) – Maximum period to consider.
- **plot_time** (`bool`) – If `True`, plot time on the x axis, else plot sampling frequency.
- **title** (`Optional[str]`) – Plot title.

Return type `Optional[BaseFigure]`

Returns Plotly figure if show is `True`, else nothing.

pytsviz.**plot_scatter_fit**(*df*, *x_col*, *y_col*, *time_col=None*, *title=None*, *fit=False*, *show=True*, ***kwargs*)

Interactive scatter plot between two time series with possibility of showing a fit line between them.

Parameters

- **df** (`DataFrame`) – Dataframe to use as the data source for the plot.
- **x_col** (`str`) – Name of the column from df to be used as x variable in the scatter plot.
- **y_col** (`str`) – Name of the column from df to be used as y variable in the scatter plot.
- **time_col** (`Optional[str]`) – Name of the column from df to be used as time variable in the plot.
- **title** (`Optional[str]`) – Plot title.
- **fit** (`Union[bool, Literal['summary']]`) – Whether to show the fit line in the plot. If set to "summary", displays fit statistics at the bottom.
- **show** (`bool`) – Whether to call `Figure.show` at the end of the function. If `False`, returns the figure.
- **kwargs** – Keyword arguments for fit, if not set to `False`. They are passed to `pandas.plot`.

Return type `Optional[BaseFigure]`

Returns Plotly figure if show is `True`, else nothing.

pytsviz.**plot_scatter_matrix**(*df*, *x_col*, *y_col=None*, *lags_x=None*, *lags_y=None*, *time_col=None*, *title=None*, *show=True*)

Interactive scatter matrix plot of a set of variables and possibly lagged versions of them.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **x_col** (str) – Name of the column to be used as first variable in the matrix.
- **y_col** (Optional[str]) – If given, name of the column to be used as second variable in the matrix.
- **lags_x** (Optional[List[int]]) – If given, list of lags to be applied to x_col and shown as lagged series in the matrix.
- **lags_y** (Optional[List[int]]) – If given, list of lags to be applied to y_col and shown as lagged series in the matrix.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_seasonal_ts**(*df*, *period*, *y_col=None*, *time_col=None*, *title=None*, *subplots=False*, *show=True*)

Interactive plot of the seasonal components of a time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **period** (Union[Literal['minute', 'hour', 'day', 'week', 'month', 'quarter', 'year'], Tuple[Callable[[Iterable], Iterable]]]) – The periodicity of the seasonal component. It can be one of 'minute', 'hour', 'day', 'week', 'month', 'quarter', 'year' or a couple of callables specifying both the periodicity and the granularity of data (e.g. daily periodicity, hourly granularity).
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **subplots** (bool) – Whether to split the seasonal components into subplots. Should be set to False if the number of components is too high.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_ts**(*df*, *y_cols=None*, *time_col=None*, *title=None*, *tf=None*, *tf_args=()*, *tf_kwargs=None*, *keep_original=True*, *show=True*)

Interactive plot of one or mode time series, with possibility of transforming them as well.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.

- **y_cols** (Optional[List[str]]) – Name of the columns from df to be used as time series values in the plot. If None, all columns are used.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **tf** (Union[Literal['Box-Cox', 'Yeo-Johnson', 'log', 'moving_average'], Callable[[Iterable], Iterable], None]) – If provided, a transformation to be applied to the time series. It can be one of "Box-Cox", "Yeo-Johnson", "log", "moving_average" or a custom callable one.
- **tf_args** (Tuple) – Positional arguments for the transformation. See tf for details.
- **tf_kwargs** (Optional[dict]) – Keyword arguments for the transformation. See tf for details.
- **keep_original** (bool) – Whether to show the original along with the transformed one. Only active if tf is not None.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_ts_analysis**(*df*, *y_col=None*, *time_col=None*, *nfft=1024*, *acf_n_lags=None*, *pacf_n_lags=None*, *alpha=0.1*, *show=True*, *title=None*)

Composite plot showing the time series, its spectral density, ACF and PACF.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **nfft** (int) – Length of the FFT used. If None the length of df will be used.
- **acf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for ACF.
- **pacf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for PACF.
- **alpha** (float) – If a number is given, the confidence intervals for the given level are displayed in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **title** (Optional[str]) – Plot title.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.**plot_ts_overview**(*df*, *y_col=None*, *n_lags=None*, *alpha=0.1*, *show=True*, *title=None*)

Composite plot showing a time series, its auto-correlation function and its distribution.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **n_lags** (Optional[int]) – Maximum number of time steps to consider as lag.
- **alpha** (float) – If a number is given, the confidence intervals for the given level are displayed in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **title** (Optional[str]) – Plot title.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

1.2.1 Submodules

pytsviz.utils module

The *utils* module contains utilities not strictly related to visualization which we often use (eg harmonics computation).

pytsviz.utils.**harmonics**(*dates*, *period*, *n*, *epoch=datetime.datetime(1900, 1, 1, 0, 0)*)

Computes harmonics for the given dates. Each harmonic is made of a couple of sinusoidal and cosinusoidal waves with frequency i/period, i = 1. . . n. The argument of the functions is the number of hours from the starting epoch.

Parameters

- **dates** (DataFrame) – A pandas series of dates.
- **period** (int) – The base period of the harmonics.
- **n** (int) – The number of harmonics to include.
- **epoch** (datetime) – The epoch used to compute the argument of the sin.

Return type DataFrame

Returns A Pandas DataFrame with dates as index and harmonics as columns.

pytsviz.viz module

The *viz* module contains functions to visualize most of the key aspects of a univariate time series such as (partial) correlograms, periodograms, line plots, . . .

pytsviz.viz.**_plot_plotly**(*df*, *kind*, *x_title=None*, *y_title=None*, *x_type=None*, *y_type=None*, ***kwargs*)

Wrapper for pandas plot function, with plotly backend and the application of the default styling template.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **kind** (str) – The kind of plot to produce.
- **x_title** (Optional[str]) – x axis title.
- **y_title** (Optional[str]) – y axis title.

- **x_type** (Optional[Literal['linear', 'log', 'date', 'category', 'multicategory']]) – If provided, enforces the x axis data type. Can be one of 'linear', 'log', 'date', 'category', 'multicategory'.
- **y_type** (Optional[Literal['linear', 'log', 'date', 'category', 'multicategory']]) – If provided, enforces the y axis data type. Can be one of 'linear', 'log', 'date', 'category', 'multicategory'.
- **kwargs** (Any) – Keyword arguments for pandas.plot.

Return type BaseFigure

Returns Plotly figure.

pytsviz.viz.**plot_acf**(*df*, *y_col=None*, *time_col=None*, *partial=False*, *n_lags=None*, *title=None*, *show_threshold=True*, *show=True*, ***kwargs*)

Interactive bar plot of the auto-correlation function of a time series up to a certain lag.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **partial** (bool) – Whether to use the partial auto-correlation function.
- **n_lags** (Optional[int]) – Maximum number of time steps to consider as lag.
- **title** (Optional[str]) – Plot title.
- **show_threshold** (bool) – Whether to compute the ACF significance threshold, defined as the 95% confidence upper bound of a time series which has no auto-correlation, and show it as a horizontal line in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_decomposed_ts**(*df*, *method*, *y_col=None*, *time_col=None*, *title=None*, *subplots=True*, *show=True*, ***decomp_kwargs*)

Interactive plot of a decomposition of a time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **method** (Literal['STL', 'seasonal_additive', 'seasonal_multiplicative']) – Method of decomposition. It can be one of "STL", "seasonal_additive" or "seasonal_multiplicative".
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **subplots** (bool) – Whether to split the decomposition components into subplots.

- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **decomp_kwargs** (Any) – Keyword arguments for the selected decomposition method. See method for details.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_distribution**(*df*, *y_col=None*, *time_col=None*, *bins=None*, *title=None*, *show=True*)

Interactive histogram of a time series. Useful to assess marginal distribution shape.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **bins** (Optional[int]) – Number of bins in the histogram.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_extended_scatter_matrix**(*df*, *time_col=None*, *y_cols=None*, *title=None*, *show=True*)

Composite plot containing scatter matrix, distributions and couple-correlations of a set of time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_cols** (Optional[List[str]]) – Name of the columns from df to be used as variables in the plot. If None, all columns are used.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_forecast**(*df*, *y_col*, *y_hat_cols*, *lower_col=None*, *upper_col=None*, *time_col=None*, *title=None*, *show=True*)

Interactive plot of a time series forecast (or multiple forecasts).

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (str) – Name of the column from df to be used as time series values in the plot.

- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **y_hat_cols** (List[str]) – Name of the columns from df to be used as forecasts for the time series in the plot.
- **lower_col** (Optional[str]) – If given, name of the column to be used as lower confidence bound for the forecast.
- **upper_col** (Optional[str]) – If given, name of the column to be used as upper confidence bound for the forecast.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_gof**(*df*, *y_col*, *y_hat_col*, *time_col=None*, *acf_n_lags=None*, *pacf_n_lags=None*, *alpha=0.1*, *title='Goodness of Fit'*, *subplot_titles=('Actual vs Predicted Series', 'Actual vs Predicted Scatter', 'Residuals', 'Residuals ACF', 'Residuals PACF')*, *show=True*)

Interactive plot of goodness of fit visualizations. In order: Actual Series vs Predicted Series, Residuals series, and Actual vs Predicted scatter plot.

Parameters

- **acf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for ACF.
- **pacf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for PACF.
- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (str) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **y_hat_col** (str) – Name of the column from df to be used as forecast for the time series in the plot.
- **alpha** (float) – If a number is given, the confidence intervals for the given level are displayed in the plot.
- **title** (str) – Plot title.
- **subplot_titles** (Tuple[str, str, str, str, str]) – Tuple of titles for each of the 3 subplots.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Tuple of plotly figures if show is True, else nothing.

pytsviz.viz.**plot_inverse_arma_roots**(*process*, *show=True*)

Plot of the inverse roots of a given ARMA process.

Parameters

- **process** (ArmaProcess) – The ARMA process (statsmodels.tsa.ArmaProcess) for which the roots are computed.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_psd**(*df*, *y_col=None*, *time_col=None*, *nfft=None*, *fs=1*, *min_period=0*, *max_period=inf*, *plot_time=False*, *title=None*, *show=True*, ***kwargs*)

Interactive histogram of the spectral density of a time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **nfft** (Optional[int]) – Length of the FFT used. If None the length of df will be used.
- **fs** (int) – Sampling frequency of df.
- **min_period** (int) – Minimum period to consider.
- **max_period** (int) – Maximum period to consider.
- **plot_time** (bool) – If True, plot time on the x axis, else plot sampling frequency.
- **title** (Optional[str]) – Plot title.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_scatter_fit**(*df*, *x_col*, *y_col*, *time_col=None*, *title=None*, *fit=False*, *show=True*, ***kwargs*)

Interactive scatter plot between two time series with possibility of showing a fit line between them.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **x_col** (str) – Name of the column from df to be used as x variable in the scatter plot.
- **y_col** (str) – Name of the column from df to be used as y variable in the scatter plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **fit** (Union[bool, Literal['summary']]) – Whether to show the fit line in the plot. If set to "summary", displays fit statistics at the bottom.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **kwargs** – Keyword arguments for fit, if not set to False. They are passed to pandas.plot.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_scatter_matrix**(*df*, *x_col*, *y_col=None*, *lags_x=None*, *lags_y=None*, *time_col=None*, *title=None*, *show=True*)

Interactive scatter matrix plot of a set of variables and possibly lagged versions of them.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **x_col** (str) – Name of the column to be used as first variable in the matrix.
- **y_col** (Optional[str]) – If given, name of the column to be used as second variable in the matrix.
- **lags_x** (Optional[List[int]]) – If given, list of lags to be applied to x_col and shown as lagged series in the matrix.
- **lags_y** (Optional[List[int]]) – If given, list of lags to be applied to y_col and shown as lagged series in the matrix.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_seasonal_ts**(*df*, *period*, *y_col=None*, *time_col=None*, *title=None*, *subplots=False*, *show=True*)

Interactive plot of the seasonal components of a time series.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **period** (Union[Literal['minute', 'hour', 'day', 'week', 'month', 'quarter', 'year'], Tuple[Callable[[Iterable], Iterable]]]) – The periodicity of the seasonal component. It can be one of 'minute', 'hour', 'day', 'week', 'month', 'quarter', 'year' or a couple of callables specifying both the periodicity and the granularity of data (e.g. daily periodicity, hourly granularity).
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **subplots** (bool) – Whether to split the seasonal components into subplots. Should be set to False if the number of components is too high.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_ts**(*df*, *y_cols=None*, *time_col=None*, *title=None*, *tf=None*, *tf_args=()*, *tf_kwargs=None*, *keep_original=True*, *show=True*)

Interactive plot of one or mode time series, with possibility of transforming them as well.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_cols** (Optional[List[str]]) – Name of the columns from df to be used as time series values in the plot. If None, all columns are used.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **title** (Optional[str]) – Plot title.
- **tf** (Union[Literal['Box-Cox', 'Yeo-Johnson', 'log', 'moving_average'], Callable[[Iterable], Iterable], None]) – If provided, a transformation to be applied to the time series. It can be one of "Box-Cox", "Yeo-Johnson", "log", "moving_average" or a custom callable one.
- **tf_args** (Tuple) – Positional arguments for the transformation. See tf for details.
- **tf_kwargs** (Optional[dict]) – Keyword arguments for the transformation. See tf for details.
- **keep_original** (bool) – Whether to show the original along with the transformed one. Only active if tf is not None.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_ts_analysis**(*df*, *y_col=None*, *time_col=None*, *nfft=1024*, *acf_n_lags=None*, *pacf_n_lags=None*, *alpha=0.1*, *show=True*, *title=None*)

Composite plot showing the time series, its spectral density, ACF and PACF.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **time_col** (Optional[str]) – Name of the column from df to be used as time variable in the plot.
- **nfft** (int) – Length of the FFT used. If None the length of df will be used.
- **acf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for ACF.
- **pacf_n_lags** (Optional[int]) – Maximum number of time steps to consider as lag for PACF.
- **alpha** (float) – If a number is given, the confidence intervals for the given level are displayed in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **title** (Optional[str]) – Plot title.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

pytsviz.viz.**plot_ts_overview**(*df*, *y_col=None*, *n_lags=None*, *alpha=0.1*, *show=True*, *title=None*)

Composite plot showing a time series, its auto-correlation function and its distribution.

Parameters

- **df** (DataFrame) – Dataframe to use as the data source for the plot.
- **y_col** (Optional[str]) – Name of the column from df to be used as time series values in the plot.
- **n_lags** (Optional[int]) – Maximum number of time steps to consider as lag.
- **alpha** (float) – If a number is given, the confidence intervals for the given level are displayed in the plot.
- **show** (bool) – Whether to call Figure.show at the end of the function. If False, returns the figure.
- **title** (Optional[str]) – Plot title.

Return type Optional[BaseFigure]

Returns Plotly figure if show is True, else nothing.

1.3 Visualization examples

1.3.1 Test

[1]:
```
import sys
sys.path.append('../../../')

import numpy as np
import pandas as pd
from numpy.random import random
import statsmodels.api as sm

from pytsviz import *
```

1.3.2 Data preparation for forecasting

Target series

[2]:
```
dataset_size=366
```

[3]:
```
t_axis = pd.date_range(start='1/1/2020', periods = dataset_size, freq="D", name="t")
```

[4]:
```
y = (1 + random(dataset_size)) * np.linspace(6, 9, dataset_size) + \
    (1 + random(dataset_size)) * np.sin(np.linspace(0, 10*np.pi, dataset_size)) + \
    (1.5 + random(dataset_size)) * np.cos(np.linspace(0, 5.4*np.pi, dataset_size))
ts = pd.Series(index=t_axis, data=y, name="y")
ts_df = ts.to_frame()
```

Features

[5]:
```
n_features = 3
```

[6]:
```
feat_matrix = random((n_features, dataset_size))
feat_df = pd.DataFrame(index=t_axis, data={f"x_{i}": feat_matrix[i] for i in range(n_
↪features)})
```

Forecast

[7]:
```
ext_t_axis = pd.date_range(start='2/1/2020', periods = dataset_size, freq="D", name="t
↪")
```

[8]:
```
y_hat = y * (1 + 0.1 * random())

forecast = pd.Series(index=ext_t_axis, data=y_hat, name="y_hat")
forecast_df = forecast.to_frame()
```

[9]:
```
c_lower = [y_hat[i] - (0.001 * i) ** 1.2 for i in range(len(y_hat))]
c_upper = [y_hat[i] + (0.001 * i) ** 1.2 for i in range(len(y_hat))]
confidence_df = pd.DataFrame(index=ext_t_axis, data={"lower confidence": c_lower,
↪"upper confidence": c_upper})
```

[10]:
```
total_df = pd.concat([ts, feat_df, forecast_df, confidence_df], axis = 1)
```

1.3.3 Time series plot

First let's plot our target time series along with one of the features we want to use with forecasting. We can also display their log transformation along with (or excluding) the originals.

[11]:
```
plot_ts(total_df, y_cols=["y", "x_0"], tf="log", tf_args = (np.e,), keep_
↪original=True)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

There are other useful transformations, as well as the possibility of passing a custom one. Here we can see the Box-Cox transform applied to our target series.

[12]:
```
plot_ts(total_df, y_cols=["y"], tf="Box-Cox", keep_original=False)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.4 Seasonal plot

We can take our target series and inspect it, looking for seasonal components. For example, let's break it down into monthly components.

[13]:
```
plot_seasonal_ts(ts_df, period="month", subplots=True)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

We can also customize the grouping of our data, by specifying a couple of function to pass as period.

These functions are used respectively to assign data points to period groups and to select the granularity with which we want to display the data.

For example, let's say we want to inspect the time series by grouping it into weeks, and keeping a daily granularity:

[14]:
```
extract_week = lambda x : x.isocalendar().week
extract_day = lambda x : x.isocalendar().day
```

[15]:
```
# noinspection PyTypeChecker
plot_seasonal_ts(ts_df, period=(extract_week, extract_day), subplots=False)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

Notice that because of the high number of weeks in our dataset we had to set subplots = False.

1.3.5 Decomposed plot

One more thing we can do is to try and apply a classic time series decomposition method, such as STL decomposition, to our series:

[16]:
```
plot_decomposed_ts(ts_df, method = "STL", subplots = True)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.6 Multiple forecasts

We can display one or more forecasts together with their target series, and also show confidence bounds by specifying their columns in the dataframe.

[17]:
```
inverted_df = pd.DataFrame(total_df.values[::-1], total_df.index, total_df.columns)

plot_forecast(total_df, y_col = "y", y_hat_cols = ["y_hat"], upper_col = "upper␣
↪confidence", lower_col = "lower confidence")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[18]:
```
y_hat_2 = pd.Series(data=y_hat*0.9, index = ext_t_axis, name = "y_hat_2")
multiple_fc_df = pd.concat([total_df, y_hat_2], axis = 1)
plot_forecast(multiple_fc_df, y_col = "y", y_hat_cols = ["y_hat", "y_hat_2"])
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.7 Scatter matrix

A useful inspection tool is the scatter matrix, that is a matrix of scatter plots between different series.

We can include features, the target as well as lagged versions of both of them.

[19]:
```
plot_scatter_matrix(total_df, x_col="x_0", y_col="y", lags_x=[5, 10, 15])
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.8 Fit

By inspecting a single pair of time series in a scatter plot we can show the fit line, together with a verbose summary of the fit computation.

[20]:
```
plot_scatter_fit(total_df, "x_0", "x_1", fit = "summary")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.014
Model:                            OLS   Adj. R-squared:                  0.012
Method:                 Least Squares   F-statistic:                     5.316
Date:                Thu, 28 Oct 2021   Prob (F-statistic):             0.0217
Time:                        15:08:49   Log-Likelihood:                -70.229
No. Observations:                 366   AIC:                             144.5
Df Residuals:                     364   BIC:                             152.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4490      0.031     14.427      0.000       0.388       0.510
x1             0.1207      0.052      2.306      0.022       0.018       0.224
==============================================================================
Omnibus:                      244.484   Durbin-Watson:                   1.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               22.749
Skew:                          -0.001   Prob(JB):                     1.15e-05
Kurtosis:                       1.779   Cond. No.                         4.38
==============================================================================
```

(continues on next page)

(continued from previous page)

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly␣
↪specified.
```

Extra arguments can be passed to modify the fit method of computation, for example:

[21]:
```
plot_scatter_fit(total_df, "x_0", "x_1",  trendline="lowess")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.9 Auto-correlation

We can produce auto-correlation (or partial auto-correlation) charts for a given time series, and optionally include both the significance correlation threshold and the confidence intervals for correlation values, to assess any significant lag in the series.

[22]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

[23]:
```
plot_acf(ts_df, show_threshold = True)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[24]:
```
plot_acf(ts_df, partial=True, alpha = 0.1, show_threshold = True)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

Another useful tool is the visualization of the distribution and the spectral decomposition of a time series.

[25]:
```
plot_distribution(total_df, bins=100, title="Distribution")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[26]:
```
plot_psd(ts_df, scaling="spectrum")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.10 Roots of an ARMA process

Finally, let's take a look at the roots of a given ARMA process; they can be used to find out is the process is stationary, among other things.

[27]:
```
arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])
ar = np.r_[1, -arparams] # add zero-lag and negate
ma = np.r_[1, maparams] # add zero-lag
arma_process = sm.tsa.ArmaProcess(ar, ma)
```

[28]:
```
plot_inverse_arma_roots(arma_process)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.3.11 Composite visualizations

To conclude, here are some common composite plots that can be produced all at once, in order to provide a comprehensive qualitative view of a (set of) time series.

[29]:
```
plot_extended_scatter_matrix(total_df.dropna(), y_cols=["y", "x_1", "x_2"])
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[30]:
```
plot_ts_overview(ts_df)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[31]:
```
plot_ts_analysis(ts_df)
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

[32]:
```
plot_gof(total_df, "y", "y_hat")
```

Data type cannot be displayed: application/vnd.plotly.v1+json, text/html

1.4 Changelog

Warning: Changes are not being tracked until a new release is made.

The change log will appear here.

1.5 Support

The best way to ask general questions about a particular use cases is to reach us on our dedicated Slack channel or email us at oss@xtreamers.io.

1.6 License

1.7 Contributor Code of Conduct

As contributors and maintainers of this project, we pledge to respect all people who contribute through reporting issues, posting feature requests, updating documentation, submitting pull requests or patches, and other activities.

We are committed to making participation in this project a harassment-free experience for everyone, regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, or religion.

Examples of unacceptable behavior by participants include the use of sexual language or imagery, derogatory comments or personal attacks, trolling, public or private harassment, insults, or other unprofessional conduct.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct. Project maintainers who do not follow the Code of Conduct may be removed from the project team.

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue or contacting one or more of the project maintainers.

This Code of Conduct is adapted from the Contributor Covenant (https://www.contributor-covenant.org), version 1.0.0, available at https://contributor-covenant.org/version/1/0/0/.

PYTHON MODULE INDEX

p

INDEX

Symbols

H

M

P