timeseries documentation

# Welcome to timeseries’s documentation![¶](#welcome-to-timeseries-s-documentation)

## Projekt[¶](#projekt)

### chart module[¶](#module-chart)

*class* chart.Chart(*df: pandas.core.frame.DataFrame*, *title: str*, *id: int*)[¶](#chart.Chart)

Bases: object

Class contains loaded time series data in form of charts.

\_\_ax *= None*[¶](#chart.Chart.__ax)

Axes of the chart.

\_\_df*: pandas.core.frame.DataFrame = None*[¶](#chart.Chart.__df)

Data frame of loaded signal.

\_\_figure*: matplotlib.backends.backend\_tkagg.FigureCanvasTkAgg = None*[¶](#chart.Chart.__figure)

Canvas of the chart.

\_\_id*: int = 0*[¶](#chart.Chart.__id)

ID of the chart.

\_\_title*: str = ''*[¶](#chart.Chart.__title)

Title of the chart.

displayChart(*display: tkinter.ttk.Frame*, *moduleDisplayer*)[¶](#chart.Chart.displayChart)

Displays chart on screen.

Parameters

* **display** – GUI component where chart should be displayed.
* **moduleDisplayer** – Method called in-line, that applies patches on chart from all active time series modules.

exportChart(*filename: str*)[¶](#chart.Chart.exportChart)

Exports chart as .png file to specified destination.

Parameters

**filename** – Directory where canvas should be saved.

getDataframe() → pandas.core.frame.DataFrame[¶](#chart.Chart.getDataframe)

Returns data frame of loaded signal.

Returns

Data frame of loaded signal.

getId() → int[¶](#chart.Chart.getId)

Returns ID of the chart.

Returns

ID of the chart.

getTitle() → str[¶](#chart.Chart.getTitle)

Returns title of the chart.

Returns

Title of the chart.

setTitle(*title: str*)[¶](#chart.Chart.setTitle)

Sets title of the chart.

Parameters

**title** – New title of the chart.

### dataGenerator module[¶](#module-dataGenerator)

*class* dataGenerator.SignalGenerator[¶](#dataGenerator.SignalGenerator)

Bases: object

Class containing sample time series signal.

\_\_dataY *= None*[¶](#dataGenerator.SignalGenerator.__dataY)

Generated signal.

\_\_genCps *= None*[¶](#dataGenerator.SignalGenerator.__genCps)

Generated change point indexes.

\_\_genMeans *= None*[¶](#dataGenerator.SignalGenerator.__genMeans)

Means in area between change points, before applying noise.

\_\_genSpreads *= None*[¶](#X0336c05831981bb220dd8c50de14d0c13ba7691)

Gaussian noise scale in area between change points.

applyPatchesToChart(*ax*)[¶](#Xd202178b3755b30377d97cea1284e2c9a431fa9)

Visualizes change points on charts’ axes.

Parameters

**ax** – Chart’s axes

generateSignal(*samples*, *cpCount*, *mean*, *meanSpread*, *noiseScale*, *noiseScaleSpread*)[¶](#X20a0549a769a7b55c83f6d4cdc1065b6f42a193)

Generates signal based on parameters. Data is stored inside the class.

Parameters

* **samples** – Number of samples (data length) in generated signal.
* **cpCount** – Number of change points in generated signal
* **mean** – Mean of all data in generated signal
* **meanSpread** – Spread of mean of all data in generated signal
* **noiseScale** – Gaussian noise scale in areas between change points in generated signal.
* **noiseScaleSpread** – Spread of gaussian noise scale in areas between change points in generated signal.

getGeneratedSignalProperties() → tuple[¶](#Xedf8e37f4ef51eda8f7fe8d27131793137a818c)

Returns properties of generated signal.

Returns

Properties of generated signal.

getSignal() → list[¶](#dataGenerator.SignalGenerator.getSignal)

Returns generated signal.

Returns

Generated signal.

### entryvalidators module[¶](#module-entryvalidators)

entryvalidators.validate\_float(*val*)[¶](#entryvalidators.validate_float)

Checks if value is float value.

Parameters

**val** – Value to check.

Returns

If val meets requirements. Rollbacks operation if false.

entryvalidators.validate\_float\_pos(*val*)[¶](#entryvalidators.validate_float_pos)

Checks if value is positive float value.

Parameters

**val** – Value to check.

Returns

If val meets requirements. Rollbacks operation if false.

entryvalidators.validate\_float\_pos\_zero(*val*)[¶](#entryvalidators.validate_float_pos_zero)

Checks if value is positive (or zero) float value.

Parameters

**val** – Value to check.

Returns

If val meets requirements. Rollbacks operation if false.

entryvalidators.validate\_int\_pos(*val*)[¶](#entryvalidators.validate_int_pos)

Checks 0 :param val: Value to check. :returns: If val meets requirements. Rollbacks operation if false.

### gui module[¶](#module-gui)

*class* gui.AnalyzerGui[¶](#gui.AnalyzerGui)

Bases: [gui.Gui](#gui.Gui)

Class contains window application of time series analyzer.

\_\_buildSectionChart()[¶](#gui.AnalyzerGui.__buildSectionChart)

Builds section with chart that displays current data.

\_\_buildSectionOutput()[¶](#gui.AnalyzerGui.__buildSectionOutput)

Builds section with output functions that display and export data.

\_\_buildSectionSettings()[¶](#gui.AnalyzerGui.__buildSectionSettings)

Builds section with settings that set properties of data.

\_\_buildSubsectionDataSelection()[¶](#X49ab66ab628d18b9f61d0872d579fc0b06ec462)

Builds subsection of settings section that allows user to choose displayed data, which was previously loaded.

\_\_buildSubsectionExportData()[¶](#X94faa78308929ad6ab4cd545248c7444ace811a)

Builds subsection of output section that allows exporting logs, chart and statistics.

\_\_buildSubsectionLoadData()[¶](#X0aa73c1081c8309598714b8b988240bfbb055b1)

Builds subsection of settings section that loads of data from memory.

\_\_buildSubsectionLogger()[¶](#gui.AnalyzerGui.__buildSubsectionLogger)

Builds subsection of output section that logs user’s actions.

\_\_buildSubsectionStatisctics()[¶](#Xfab2d8901414aa2589c819b84fecdd47d0ea0a2)

Builds subsection of output section that displays data of modules in tabular form.

\_\_buildSubsectionTSModules()[¶](#Xaa3faa098dba410eb76c0a7364ec53f618dbbd7)

Builds subsection of settings section that allows user to choose displayed time series analyzing modules.

\_\_chartId*: int = -1*[¶](#gui.AnalyzerGui.__chartId)

ID of currently displayed chart

\_\_clearSectionChart()[¶](#gui.AnalyzerGui.__clearSectionChart)

Method clears the view of section with chart.

\_\_deleteChart()[¶](#gui.AnalyzerGui.__deleteChart)

Deletes the current viewed chart.

\_\_hubCharts*:* [*hubCharts.HubCharts*](index.html#hubCharts.HubCharts) *= <hubCharts.HubCharts object>*[¶](#gui.AnalyzerGui.__hubCharts)

Functional component with methods concerning loaded data in form of charts.

\_\_hubImporters*:* [*hubImporters.HubImporters*](index.html#hubImporters.HubImporters) *= <hubImporters.HubImporters object>*[¶](#gui.AnalyzerGui.__hubImporters)

Functional component with methods concerning data importers.

\_\_hubTSModules*:* [*hubTSModules.HubTSModules*](index.html#hubTSModules.HubTSModules) *= <hubTSModules.HubTSModules object>*[¶](#gui.AnalyzerGui.__hubTSModules)

Functional component with methods concerning time series analyzing modules.

\_\_importer*:* [*importers.baseimport.ImportMethod*](index.html#importers.baseimport.ImportMethod) *= None*[¶](#gui.AnalyzerGui.__importer)

Chosen by user importer function (ex. .xls import when loading microsoft excel spreadsheet).

\_\_refreshTreeviewStatistics()[¶](#X14c1efc8f43a894aad77541a89404b4befedbcd)

Removes the content of combobox and table view with all active time series modules.

\_\_sectionChart*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__sectionChart)

Section with chart that displays current data.

\_\_sectionOutput*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__sectionOutput)

Section with output functions that display and export data.

\_\_sectionSettings*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__sectionSettings)

Section with settings that set properties of data.

\_\_subsectionDataSelection*: tkinter.Frame = None*[¶](#X28ed44889e3c8060cc66da395f6d9f7de4f059b)

Subsection of settings section that allows user to choose displayed data, which was previously loaded.

\_\_subsectionDataSelection\_buttonDeleteData\_action()[¶](#Xe3c48655266fd81a81ff53f8951b9a4b8ce37ea)

Method called when activating delete button in subsection of settings section that allows user to choose displayed data, which was previously loaded. Deletes selected data.

\_\_subsectionDataSelection\_buttonUpdateData\_action()[¶](#X502b7c78ec94a787c5c91c844de932d5b9ce42d)

Method called when activating update button in subsection of settings section that allows user to choose displayed data, which was previously loaded. Updates chart displays with chosen data.

\_\_subsectionDataSelection\_chartscb\_action(*event*)[¶](#Xce2120d7b8bb90b06d2b6f3227ba6e17ffd8333)

Method called when updating charts combobox in subsection of settings section that allows user to choose displayed data, which was previously loaded. Updates the preview of selected data.

\_\_subsectionExportData*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__subsectionExportData)

Subsection of output section that allows exporting logs, chart and statistics.

\_\_subsectionExportData\_buttonExportChart\_action()[¶](#X23bdd01766588c8d31ab29b23e038575909b776)

Method called when activating export button in subsection of output section that allows exporting logs, chart and statistics Exports displayed chart to image format.

\_\_subsectionExportData\_buttonExportLogs\_action()[¶](#X0789a54709dad91e0543ea692cbb8e8300b704a)

Method called when activating export button in subsection of output section that allows exporting logs, chart and statistics Exports logs to text format.

\_\_subsectionExportData\_buttonExportStats\_action()[¶](#X3ae0c94ed0a9ae689783aa1f22f5fab9066ab0d)

Method called when activating export button in subsection of output section that allows exporting logs, chart and statistics Exports output modules data to text format.

\_\_subsectionLoadData*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__subsectionLoadData)

Subsection of settings section that loads of data from memory.

\_\_subsectionLoadData\_buttonImportDataFromFile\_action()[¶](#Xecede9f27b7c277738b876c1a6148c577158ce8)

Method called when activating export button in subsection of settings section that loads of data from memory. Exports data from importer to application’s memory.

\_\_subsectionLoadData\_buttonImportData\_action()[¶](#X55bcd38e55e2b6487041a90145c3c32fedeff84)

Method called when activating import button in subsection of settings section that loads of data from memory. Configures chosen importer function.

\_\_subsectionLogger*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__subsectionLogger)

Subsection of output section that logs user’s actions.

\_\_subsectionStatisctics*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__subsectionStatisctics)

Subsection of output section that displays data of modules in tabular form.

\_\_subsectionStatisctics\_comboboxOutputModules\_action(*event*)[¶](#X54151b03aea670defed0b732e2ff31d969e833c)

Method called when updating modules combobox in subsection of output section that displays data of modules in tabular form. Rebuilds a table view of analyzing module data.

\_\_subsectionTSModules*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.__subsectionTSModules)

Subsection of settings section that allows user to choose displayed time series analyzing modules.

\_\_subsectionTSModules\_comboboxTSModule\_action(*event*)[¶](#X990c9f9a029474564085d129f6df31734cf6d14)

Method called when updating modules combobox in subsection of settings section that allows user to choose displayed time series analyzing modules. Rebuilds a menu with module parameters.

\_\_updateChart()[¶](#gui.AnalyzerGui.__updateChart)

Updates the view of section with chart.

\_\_updateComboboxCharts()[¶](#gui.AnalyzerGui.__updateComboboxCharts)

Refreshes the content of combobox with all loaded time series modules.

\_buildSections()[¶](#gui.AnalyzerGui._buildSections)

Constructs all sections of an application.

*class* gui.GeneratorGui[¶](#gui.GeneratorGui)

Bases: [gui.Gui](#gui.Gui)

\_\_buildSectionChart()[¶](#gui.GeneratorGui.__buildSectionChart)

Builds section with chart that displays current generated data.

\_\_buildSectionSettings()[¶](#gui.GeneratorGui.__buildSectionSettings)

Builds section with settings that set properties of generated data.

\_\_buildSubSectionChartGenerator()[¶](#X1ffdb2f054c4673e4a3e9bc48f1650a4cf3024d)

Builds subsection of settings section that configures parameters of generated chart.

\_\_buildSubSectionExportData()[¶](#Xf9092e1324b6eac036d515cf012111d475f159b)

Builds subsection of settings section that allows exporting generated chart and statistics.

\_\_cpCount *= 2*[¶](#gui.GeneratorGui.__cpCount)

Number of change points in generated signal.

\_\_genDf *= Empty DataFrame Columns: [] Index: []*[¶](#gui.GeneratorGui.__genDf)

Data frame of generated signal.

\_\_outvarCPSpread *= None*[¶](#gui.GeneratorGui.__outvarCPSpread)

Gaussian noise scale in areas between change points in generated signal.

\_\_outvarMean *= None*[¶](#gui.GeneratorGui.__outvarMean)

Mean of all data in generated signal.

\_\_outvarSpread *= None*[¶](#gui.GeneratorGui.__outvarSpread)

Spread of mean of all data in generated signal.

\_\_outvarSpreadCPSpread *= None*[¶](#gui.GeneratorGui.__outvarSpreadCPSpread)

Spread of gaussian noise scale in areas between change points in generated signal.

\_\_samples *= 100*[¶](#gui.GeneratorGui.__samples)

Number of samples (data length) in generated signal.

\_\_sectionChart*: tkinter.Frame = None*[¶](#gui.GeneratorGui.__sectionChart)

Section with chart that displays current generated data.

\_\_sectionSettings*: tkinter.Frame = None*[¶](#gui.GeneratorGui.__sectionSettings)

Section with settings that set properties of generated data.

\_\_sectionsubChartGenerator*: tkinter.Frame = None*[¶](#Xd9cb1431f49a00fa5faa28b1a76fc74766852e4)

Subsection of settings section that configures parameters of generated chart.

\_\_signalGenerator*:* [*dataGenerator.SignalGenerator*](index.html#dataGenerator.SignalGenerator) *= <dataGenerator.SignalGenerator object>*[¶](#gui.GeneratorGui.__signalGenerator)

\_\_subsectionExportData*: tkinter.Frame = None*[¶](#gui.GeneratorGui.__subsectionExportData)

Subsection of settings section that allows exporting generated chart and statistics.

\_\_subsectionExportData\_buttonExport\_action()[¶](#Xef6e9191088702c713859191d683f620a90dd57)

Method called when activating export button in subsection of settings section that allows exporting generated chart and statistics. Exports generated data to text format and csv spreadsheet.

\_\_subsectionExportData\_buttonUpdateData\_action()[¶](#X1f96296974edaec9d023cb9f1903b992a404ec7)

Method called when activating export button in subsection of settings section that allows exporting generated chart and statistics. Generates a new, randomized signal.

\_buildSections()[¶](#gui.GeneratorGui._buildSections)

Constructs all sections of an application.

*class* gui.Gui[¶](#gui.Gui)

Bases: object

Base class responsible for constructing window application.

\_\_root *= None*[¶](#gui.Gui.__root)

Reference to root window.

\_buildSections()[¶](#gui.Gui._buildSections)

Method meant to be overwritten. Constructs all sections of an application.

buildGui()[¶](#gui.Gui.buildGui)

Constructs and runs root window for an application.

getRoot()[¶](#gui.Gui.getRoot)

### hubCharts module[¶](#module-hubCharts)

*class* hubCharts.HubCharts[¶](#hubCharts.HubCharts)

Bases: object

Class with methods concerning loaded data in form of charts.

\_\_allCharts *= None*[¶](#hubCharts.HubCharts.__allCharts)

List containing loaded data.

\_\_id *= 0*[¶](#hubCharts.HubCharts.__id)

ID that will be assigned to next generated chart.

addChart(*df: pandas.core.frame.DataFrame*, *title: str*)[¶](#hubCharts.HubCharts.addChart)

Adds chart to list.

Parameters

* **df** – Data frame from which charty will be built.
* **sectititleon** – Title of chart.

displayChart(*chartId: int*, *section: tkinter.Frame*, *moduleDisplayer*)[¶](#hubCharts.HubCharts.displayChart)

Displays chart in specified location.

Parameters

* **chartId** – ID of chart to display.
* **section** – GUI component where chart should be displayed.
* **moduleDisplayer** – Method called in-line, that applies patches on chart from all active time series modules.

getAllChartDisplayNames()[¶](#Xa9b54f3b60f8e75a5ad5b33c59b010e0ad14e51)

Returns display names for all loaded charts.

Returns

Requested list.

getAllCharts()[¶](#hubCharts.HubCharts.getAllCharts)

getChartById(*chartId: int*) → [chart.Chart](index.html#chart.Chart)[¶](#hubCharts.HubCharts.getChartById)

Returns chart of argument’s id. Names should be unique in order to find a single instance

Parameters

**chartId** – ID of the chart.

Returns

Requested list.

getChartIdByDisplayName(*name: str*) → int[¶](#Xa178dcb047cc5711ac62a3b99a94a2af7cb94f2)

Returns id of chart with given display name.

Parameters

**name** – Display name, with format ‘{chart title} ({chart id})’.

Returns

Requested list.

removeChart(*chartId: int*)[¶](#hubCharts.HubCharts.removeChart)

Removes chart from list.

Parameters

**chartId** – ID of chart to remove.

### hubImporters module[¶](#module-hubImporters)

*class* hubImporters.HubImporters[¶](#hubImporters.HubImporters)

Bases: object

Class with methods concerning data importers.

\_\_allDrivers *= None*[¶](#hubImporters.HubImporters.__allDrivers)

List containing all import methods.

\_\_allMethods *= None*[¶](#hubImporters.HubImporters.__allMethods)

List containing all import method names.

addMethod(*method:* [*importers.baseimport.ImportMethod*](index.html#importers.baseimport.ImportMethod))[¶](#hubImporters.HubImporters.addMethod)

Adds file extension method to list.

Parameters

**module** – Import method to add.

getAllDrivers()[¶](#hubImporters.HubImporters.getAllDrivers)

getAllMethods()[¶](#hubImporters.HubImporters.getAllMethods)

getImporterByFileextension(*fileextension: str*) → [importers.baseimport.ImportMethod](index.html#importers.baseimport.ImportMethod)[¶](#X510343f014b9d8225373e5f50b2e3e5fc9e6314)

Returns import method based on operated file extension.

Parameters

**fileextension** – File extension of operated method.

Returns

Requested method.

loadImporters() → None[¶](#hubImporters.HubImporters.loadImporters)

Loads the list with all provided importers.

### hubTSModules module[¶](#module-hubTSModules)

from tkinter import \* from tkinter import ttk import numpy as np import matplotlib.patches as patches import math import colorsys import statsmodels.api as sm from pandas.core.frame import DataFrame import pandas as pd

*class* hubTSModules.HubTSModules[¶](#hubTSModules.HubTSModules)

Bases: object

Class with methods concerning loaded data in form of charts.

\_\_allModules *= None*[¶](#hubTSModules.HubTSModules.__allModules)

List containing all loaded time series modules.

addModule(*module:* [*tsmodules.basemod.TSModule*](index.html#tsmodules.basemod.TSModule))[¶](#hubTSModules.HubTSModules.addModule)

Adds time series module to list.

Parameters

**module** – Time series module to add.

displayModules(*ax*, *plotdf*)[¶](#hubTSModules.HubTSModules.displayModules)

Displays all selected modules’ patches on selected plot.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

getAllActiveModuleNames()[¶](#X311bd7b7a449a34ba0b89ea36fa8c51c54d0806)

Returns all active time series module names.

Returns

Requested list.

getAllActiveModules()[¶](#Xa0223d4967499a19ce4700f3f3765f127b17bbf)

Returns all active time series modules.

Returns

Requested list.

getAllModuleNames()[¶](#X3ddf606038d4c368c7f5f14dc6216e0d0eda90b)

Returns all module names.

Returns

All module names.

getAllModules()[¶](#hubTSModules.HubTSModules.getAllModules)

getModuleByName(*name: str*) → [tsmodules.basemod.TSModule](index.html#tsmodules.basemod.TSModule)[¶](#X9cdd5a976cf3f922caa926ec2b3dd18fac28a04)

Returns module of argument’s name.

Parameters

**name** – Name of the module.

Returns

Requested module.

loadModules()[¶](#hubTSModules.HubTSModules.loadModules)

Loads the list with all provided modules.

### importers package[¶](#importers-package)

#### Submodules[¶](#submodules)

#### importers.baseimport module[¶](#module-importers.baseimport)

*class* importers.baseimport.ImportMethod(*extension*, *filedialogTitle*, *defaultTitle*)[¶](#importers.baseimport.ImportMethod)

Bases: object

Base class responsible for loading time series data from different files.

\_\_defaultTitle*: str = ''*[¶](#Xfd29bf2a22c072af03279416c443f30c5e48cee)

Default, preffered title of dataset, which should be applied after extracting data.

\_\_directory*: str = ''*[¶](#X64e11a3c411790a8070c82642c242b636ed2144)

Directory where data is located.

\_\_extension*: str = ''*[¶](#X209b8cecb12cd8faada16f6b6884a5d4928d66e)

Supported format of import method.

\_\_filedialogTitle*: str = ''*[¶](#Xd7b1f474414c0d8b6c9de70b832b2e0372e665a)

Format of import method displayed in file dialog.

extractData() → pandas.core.frame.DataFrame[¶](#X25348833775265980f08ca3a1445cd783520075)

Method meant to be overwritten. Extracts signal from provided data.

Returns

Data frame of single, selected time series.

getDefaultTitle() → str[¶](#X7098ca5d73170f3b8696f7d68b6eae62896d7fa)

Returns default, preffered title of dataset, which should be applied after extracting data.

Returns

Default, preffered title of dataset, which should be applied after extracting data.

getDirectory() → str[¶](#X3c30f3974b6a820134206efa967f2eb20c8686f)

Returns directory where data is located.

Returns

Directory where data is located.

getExtension() → str[¶](#X0c638a5e1954c22fbe5a1ccc5a684a74b808631)

Returns supported format of import method.

Returns

Format of import method displayed in file dialog.

getFiledialogTitle() → str[¶](#Xf0b7cfb4198c0309c9b74c2460d70d6e976fca9)

Returns format of import method displayed in file dialog.

Returns

Format of import method displayed in file dialog.

importSettingsGui(*section: tkinter.ttk.Frame*)[¶](#Xaa4a9e5d61133ae7265e23a5d4bc99a026b98a2)

Method meant to be overwritten. Provides GUI elements for selection of specific signal from provided data.

Parameters

**section** – GUI component where chart should be displayed.

setDirectory(*directory*)[¶](#Xfdd90370d89eca259e26354bd9916842fa21118)

Sets new directory where data should be located.

Parameters

**directory** – New directory where data should be located.

#### importers.csvimport module[¶](#module-importers.csvimport)

*class* importers.csvimport.CsvImportMethod(*extension*, *filedialogTitle*, *defaultTitle*)[¶](#importers.csvimport.CsvImportMethod)

Bases: [importers.baseimport.ImportMethod](#importers.baseimport.ImportMethod)

Class responsible for loading time series data from .csv files.

\_\_datax *= None*[¶](#X1d7ac818371cb956684e4b32d6fd3fc382a74d2)

Loaded X-axis data. Should contain indexes with constant intervals.

\_\_datay *= None*[¶](#X40e85163cb7bef7ac4ef33113f2a86d796c0a8f)

Loaded Y-axis data. Should data of observed phenomenon.

\_\_defaultTitle*: str*[¶](#Xd3f0c14674c6ea052282caf43dd04446c1ce710)

Default, preffered title of dataset, which should be applied after extracting data.

\_\_directory*: str*[¶](#Xd5620863ea3c743d4826c54fcfe74e3344153bf)

Directory where data is located.

\_\_extension*: str*[¶](#Xced3bebb55387d0bcf674cf9316ec6dfe096fd5)

Supported format of import method.

\_\_filedialogTitle*: str*[¶](#Xb68027301dff72e0c4fa35ac9314a57a01c914a)

Format of import method displayed in file dialog.

\_\_xSeries *= None*[¶](#X6461abced62e10ae8888df29ef1d9a16539f292)

Chosen X-axis name from chosen sheet

\_\_xSeriesProvided *= False*[¶](#Xe3e7b2ca7207c8ec0eadcbde62fdff3f7deb37c)

Auxiliary variable, determines if user provided X-axis data.

\_\_ySeries *= None*[¶](#X94392d4186f0c27a544def2e51929bff3ef3a67)

Chosen Y-axis name from chosen sheet.

\_\_ySeriesProvided *= False*[¶](#X72b6c5069e6618d41e36b55bbe35e3881163086)

Auxiliary variable, determines if user provided Y-axis data.

extractData() → pandas.core.frame.DataFrame[¶](#Xab5dbd36fdc8b8b7328fbbc2fe600c27d4b21f7)

Extracts signal from provided data.

Returns

Data frame of single, selected time series.

importSettingsGui(*section: tkinter.ttk.Frame*)[¶](#X97021c1ba96de3c078c3464efe233089e9a69f5)

Provides GUI elements for selection of specific signal from provided data.

onXYprovided()[¶](#X47233659772b8c663289eca7ec101d6ebf4c4c3)

Determines behaviour of import method’s GUI behaviour when X and Y axis data are provided.

onXchange(*event*)[¶](#X4a10f8747c4be186ebb10d2326d81759728f7b0)

Determines behaviour of import method’s GUI behaviour when X axis data is provided.

onYchange(*event*)[¶](#X5fc1f88460f7dbd8e17e0391f1121ebe328b6bd)

Determines behaviour of import method’s GUI behaviour when Y axis data is provided.

#### importers.xlsimport module[¶](#module-importers.xlsimport)

*class* importers.xlsimport.XlsImportMethod(*extension*, *filedialogTitle*, *defaultTitle*)[¶](#importers.xlsimport.XlsImportMethod)

Bases: [importers.baseimport.ImportMethod](#importers.baseimport.ImportMethod)

Class responsible for loading time series data from .xls files.

\_\_datax *= []*[¶](#X4298a020bae7e61132c05da6e8d4e524d49d24b)

Loaded X-axis data. Should contain indexes with constant intervals.

\_\_datay *= []*[¶](#X7fbc90d2c7e953412197eab1c0f85a5553f10a6)

Loaded Y-axis data. Should data of observed phenomenon.

\_\_defaultTitle*: str*[¶](#Xacac4733323813e85f60bb2e41e0bf66bc70916)

Default, preffered title of dataset, which should be applied after extracting data.

\_\_directory*: str*[¶](#Xb28d6ad95f092af905609917ee3d9d2fa109957)

Directory where data is located.

\_\_extension*: str*[¶](#Xc939403272ab3c0e30fa228a38e7edc7a49d93e)

Supported format of import method.

\_\_filedialogTitle*: str*[¶](#X0766e6d40028085c4b633f868110a5fa8ac5eaf)

Format of import method displayed in file dialog.

\_\_sheet *= None*[¶](#Xadbaaad074fa0ee66a829a42c0873ae6b8a8457)

Chosen sheet name from spreadsheet.

\_\_xSeries *= None*[¶](#X3807cfc7fd687b87d2294f7d59496bea0fe9760)

Chosen X-axis name from chosen sheet

\_\_xSeriesProvided *= False*[¶](#X6ceae5e49bac2fd35f19b7cb39d2a8d4de6f418)

Auxiliary variable, determines if user provided X-axis data.

\_\_ySeries *= None*[¶](#X3eca5fe234b9f65bea6a225f87c5ec285d80b3d)

Chosen Y-axis name from chosen sheet.

\_\_ySeriesProvided *= False*[¶](#X1a83d9af54cf5d54365b55c062ad439c183475c)

Auxiliary variable, determines if user provided Y-axis data.

extractData() → pandas.core.frame.DataFrame[¶](#X896637b0af50ffe1a553408e3654769464c7b79)

Extracts signal from provided data.

Returns

Data frame of single, selected time series.

importSettingsGui(*section: tkinter.ttk.Frame*)[¶](#Xe106966a257be5302003c60da5a3fa72bdc85b8)

Provides GUI elements for selection of specific signal from provided data.

onSheetChange(*event*)[¶](#X826e2d14a86c278f32854ad1b4914a3df0141a7)

Determines behaviour of import method’s GUI behaviour when sheet is chosen.

onXYprovided()[¶](#Xf47ac7472330a6858ed15566686a0dc7eea20ab)

Determines behaviour of import method’s GUI behaviour when X and Y axis data are provided.

onXchange(*event*)[¶](#Xa5d81dc5dc345de0f5d2b298c95b5385458e3bf)

Determines behaviour of import method’s GUI behaviour when X axis data is provided.

onYchange(*event*)[¶](#X794e0dfde8a1d7cfeecac890fcbe02e7c56593f)

Determines behaviour of import method’s GUI behaviour when Y axis data is provided.

#### Module contents[¶](#module-importers)

### loader module[¶](#module-loader)

loader.conf *= None*[¶](#loader.conf)

List with application configurations.

loader.dirs *= None*[¶](#loader.dirs)

List with directories.

loader.lang *= None*[¶](#loader.lang)

List with language labels.

loader.loadConfig(*language: str*) → bool[¶](#loader.loadConfig)

Loads with configuration lists from config.yml file.

Parameters

**language** – Language to load.

Returns

Should program be started.

### logger module[¶](#module-logger)

logger.log(*text: str*)[¶](#logger.log)

Logs a message to a component.

Parameters

**text** – Text to log.

logger.win*: tkinter.Text = None*[¶](#logger.win)

Text component where logs are stored.

### rupturesefficiency module[¶](#module-rupturesefficiency)

*class* rupturesefficiency.RupturesEfficiency[¶](#rupturesefficiency.RupturesEfficiency)

Bases: object

Class evaluating efficiency of implemented change point detection algorithms. It has no gui, it’s intended to be displayed in command prompt.

evaluate(*settrue: list*, *seteval: list*, *m: float*) → tuple[¶](#X827f52ec8a343c104a6abf7b50243094f7e47c2)

Evaluates change point detection results.

Parameters

* **settrue** – Set of true change points.
* **seteval** – Set of estimated change points.
* **m** – Precision’s and recall’s margin argument.

Returns

User input.

f *= None*[¶](#rupturesefficiency.RupturesEfficiency.f)

Handle to file, where results should be stored.

outputCPs(*header: str*, *res: list*)[¶](#X96c0d88e32cc81a1f52962c65971b1c03f811bd)

Writes results of change point detection algorithms in single iteration.

Parameters

* **header** – Display that helps the user read final file.
* **res** – List of change points

outputSummary(*header: str*, *eval:* [*rupturesefficiency.RupturesEvaluations*](index.html#rupturesefficiency.RupturesEvaluations))[¶](#Xb685f0ad53563b2fda5efcd8b58ccbe688999ba)

Writes results of evaluation metrics

Parameters

* **header** – Display that helps the user read final file.
* **eval** – Class concatenating evaluations of change point finding algorithms.

provideFloat(*display: str*) → float[¶](#X615144850411f2b2ad4a2de3e1fecde55893f68)

Asks user for input in form of float value.

Parameters

**display** – Display that helps the user what he is about to put.

provideFromMap(*display: str*, *dict: map*) → str[¶](#X0cd82bfc2a278d4de0f67b412107041a0856c23)

Asks user for input in form of single occurence from the map.

Parameters

* **display** – Display that helps the user what he is about to put.
* **dict** – Map with possible values.

Returns

User input.

providePositiveBool(*display: str*) → bool[¶](#X5a5cb7d625174fce124246cf4f926759ea65eb4)

Asks user for input in form of bool value.

Parameters

**display** – Display that helps the user what he is about to put.

Returns

User input.

providePositiveFloat(*display: str*) → float[¶](#Xa26a0fe856159d8608ae8bb2c0401eed194730e)

Asks user for input in form of positive float value.

Parameters

**display** – Display that helps the user what he is about to put.

Returns

User input.

providePositiveFloatOrZero(*display: str*) → float[¶](#X2c5aefd238fb718ffd164484b6678bca14610a7)

Asks user for input in form of positive float (or zero) value.

Parameters

**display** – Display that helps the user what he is about to put.

Returns

User input.

providePositiveInt(*display: str*) → int[¶](#Xe04ecf551704d4987c0894973fdb5d92ffac795)

Asks user for input in form of positive int value.

Parameters

**display** – Display that helps the user what he is about to put.

Returns

User input.

run()[¶](#Xe707cf7a4a9efed083b4b5b06e5930fa81e8cee)

Runs evaluation of efficiency of implemented change point detection algorithms.

*class* rupturesefficiency.RupturesEvaluations[¶](#rupturesefficiency.RupturesEvaluations)

Bases: object

Class concatenating evaluations of change point finding algorithms.

appendAll(*\*args*)[¶](#Xf1a003bf6722b8abfc5fe821ad8c24dac5ec544)

Adds a record to metrics arrays

Parameters

**args** – Array of metrics to add.

hs *= None*[¶](#Xad33d8c9655f687eeeece49195cab5273fe9f0d)

List of Hausdorff Metrics results.

l *= None*[¶](#rupturesefficiency.RupturesEvaluations.l)

Estimated:True change point count ratio

ps *= None*[¶](#X74e2e056d85d1e1e32c05e403f33569305ca213)

List of Precision Metrics results.

ris *= None*[¶](#Xc2b1d8aaed03aeeff807cca4dad6aade19d9941)

List of Rand index Metrics results.

rs *= None*[¶](#Xb27cffe1f135e135ed600ab4dcf255577472c6d)

List of Recall Metrics results.

### tsmodules package[¶](#tsmodules-package)

#### Submodules[¶](#submodules)

#### tsmodules.autocorrelation module[¶](#module-tsmodules.autocorrelation)

*class* tsmodules.autocorrelation.AutocorrelationsModule(*\_TSModule\_\_name*)[¶](#Xa4766e1e4cce20d88b7f69fc80aafbec2cdeac3)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing autocorrelation and partial autocorrelation time series module.

\_\_outvarLag *= None*[¶](#X5f42aaee07e6042711c9f424eab87ec054271de)

The length of the lag.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#Xa8beeff0ce718e4bd963408af751e48bb6087e8)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#X961f5e33d05e7e50a9056914b8819e9ec507115)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.basemod module[¶](#module-tsmodules.basemod)

*class* tsmodules.basemod.TSModule(*\_TSModule\_\_name*)[¶](#tsmodules.basemod.TSModule)

Bases: object

Base class for implementation of single time series module.

\_\_dictColor *= None*[¶](#tsmodules.basemod.TSModule.__dictColor)

Dictionary of colors, that can be used to display module’s data.

\_\_name *= 'Unnamed Module'*[¶](#tsmodules.basemod.TSModule.__name)

Name of time series module.

\_\_outvarActive *= None*[¶](#Xd144a1d0b3acc7848201000e760de9f99b2100a)

Should this module be considered in time series data analysis.

\_\_outvarColor *= None*[¶](#tsmodules.basemod.TSModule.__outvarColor)

Chosen color used to display module’s data.

buildCheckbox(*section: tkinter.ttk.Frame*)[¶](#tsmodules.basemod.TSModule.buildCheckbox)

Provides GUI elements for time series module activation.

Parameters

**section** – GUI component where module activation should be displayed.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#tsmodules.basemod.TSModule.buildConfig)

Method meant to be overwritten. Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

buildMenu(*section: tkinter.ttk.Frame*)[¶](#tsmodules.basemod.TSModule.buildMenu)

Provides GUI elements for time series module items.

Parameters

**section** – GUI component where module items should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#tsmodules.basemod.TSModule.displayModule)

Method meant to be overwritten. Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

getDisplayColor() → str[¶](#Xa687475a27f58f2c32ca593be26857716d41cfd)

Returns color used to display module’s data.

Returns

Color used to display module’s data.

getModuleName() → str[¶](#tsmodules.basemod.TSModule.getModuleName)

Returns name of time series module.

Returns

Name of time series module.

isModelActive() → str[¶](#tsmodules.basemod.TSModule.isModelActive)

Returns true if this module be considered in time series data analysis.

Returns

Should this module be considered in time series data analysis.

outputDf *= Empty DataFrame Columns: [] Index: []*[¶](#tsmodules.basemod.TSModule.outputDf)

Data frame with processed time series module data, that was displayed on chart’s axes, or that couldn’t be displayed with line graph.

#### tsmodules.decompose module[¶](#module-tsmodules.decompose)

*class* tsmodules.decompose.DecomposeModule(*name*)[¶](#tsmodules.decompose.DecomposeModule)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing decomposition of time series.

\_\_allComponents *= None*[¶](#Xe4d2a5ce8d4b4caa99684980bfa86a308949bb0)

Components of explanatory parts of time series.

\_\_modelDict *= None*[¶](#X4fb187d8c8aa6589613abc1c643fe30e2e396ef)

Dictionary of decompose models, that can be used in seasonal decomposition function.

\_\_outvarLag *= None*[¶](#X6ba3597c0fce299f465d7232ff2d7df63385ddb)

The length of the lag.

\_\_outvarModel *= None*[¶](#Xe4e73770e7811ccc4dc792d5909a565b55f2d77)

Model of time series to decompose.

\_\_selectedComponents *= None*[¶](#X2877578a6d5f995372fa0ed01677f11a97d558b)

Selected components of explanatory parts of time series.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#X40fa4d9d1c095ec2eee9f55cb50a458cf124bb1)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#X01fe8b14416c92693b779bcd67efd649c7897ab)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.differentation module[¶](#module-tsmodules.differentation)

*class* tsmodules.differentation.DifferentiationModule(*\_TSModule\_\_name*)[¶](#X672b172bcab45b08cd28bee7ac25fdbf96179a2)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing differentiation time series module.

\_\_outvarLag *= None*[¶](#X7911c30ede638e5b3ab56d1bebbbfcbabbc65cc)

The length of the lag.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#X8869dd06642b881159404f54f32d6fa04511ad3)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#Xc3c51b31dedae6f81924640370f392eb8ea89e3)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.originaldata module[¶](#module-tsmodules.originaldata)

*class* tsmodules.originaldata.OriginalDataDisplayModule(*\_TSModule\_\_name*)[¶](#Xc8af7577ff4873d4734365c097eda15e965bbf8)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing original data display time series module.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#Xd37f55c06157d97f7a75d915af7aee55df3e0bb)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.rollingmean module[¶](#module-tsmodules.rollingmean)

*class* tsmodules.rollingmean.RollingMeanModule(*\_TSModule\_\_name*)[¶](#tsmodules.rollingmean.RollingMeanModule)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing rolling mean time series module.

\_\_rollCount *= 5*[¶](#X20206611e3829af87448816df7ab3ebaacf9371)

Quanity of rolling value.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#X501a9b1f6927af3606a5e975fbf0c7cd02811ca)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#X6ddce112c58cd1d9832c5205a18001794933f98)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.rollingstd module[¶](#module-tsmodules.rollingstd)

*class* tsmodules.rollingstd.RollingStdModule(*\_TSModule\_\_name*)[¶](#tsmodules.rollingstd.RollingStdModule)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing rolling standard deviation time series module.

\_\_rollCount *= 5*[¶](#Xf0ad7c4a092f2d4537c32f3cc920a757983b4af)

Quanity of rolling value.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#X42158418eb508b3213b72fc1795b21d672fb955)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#Xde210be89f2793ea039c727c55f794112a42270)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.ruptureeval module[¶](#module-tsmodules.ruptureeval)

*class* tsmodules.ruptureeval.RupturesEvaluation(*\_TSModule\_\_name*)[¶](#tsmodules.ruptureeval.RupturesEvaluation)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing change point detection evaluation in time series modules. Evaluations include Hausdorff metric, rand index, precision and recall.

\_\_argError()[¶](#Xae16565e81bd8d9b9d131aeb1213d8b510a5e3d)

Method displayed when sets have been wrongly filled.

\_\_outvarEstset *= None*[¶](#X5baeef02148a175cb5163f37e5a3bbec893c368)

Second (predicted) set of change point indexes.

\_\_outvarMargin *= None*[¶](#X22dd0796c5ff9ee43cfeee4858533b53ff96625)

Margin value used to calculate true positives.

\_\_outvarOrigset *= None*[¶](#X4e6a711f5c7cee7d66cee64cbd0df7919603b79)

First (original) set of change point indexes.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#X72e9d7dc8ebecbdc8ae4b72f286f6470d37009d)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#Xe410bb0cea2ad8ff9ac74ef2a441a4859630497)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

#### tsmodules.rupturesmods module[¶](#module-tsmodules.rupturesmods)

*class* tsmodules.rupturesmods.RupturesBinsegModule(*name*)[¶](#Xac9a618dadf833e0c15e897f8f0794b5517313b)

Bases: [tsmodules.rupturesmods.RupturesModule](#tsmodules.rupturesmods.RupturesModule)

Class implementing Binary segmentaton algorithm that detects change points.

calculateChangePoints(*signal*) → list[¶](#Xf2cd32349a2ecf0afc5353430d2cbfbff83503e)

Calculates change point indexes.

Parameters

**signal** – Time series data.

Returns

Indexes of change points.

*class* tsmodules.rupturesmods.RupturesBottomUpModule(*name*)[¶](#X1b99cb8be733faf831ed1b448f0a03f8045459d)

Bases: [tsmodules.rupturesmods.RupturesModule](#tsmodules.rupturesmods.RupturesModule)

Class implementing Bottom-Up algorithm that detects change points.

calculateChangePoints(*signal*) → list[¶](#Xb2d6e536c8bed29cd7dc6c617621b5c66f2535f)

Calculates change point indexes.

Parameters

**signal** – Time series data.

Returns

Indexes of change points.

*class* tsmodules.rupturesmods.RupturesModule(*name*)[¶](#tsmodules.rupturesmods.RupturesModule)

Bases: [tsmodules.basemod.TSModule](#tsmodules.basemod.TSModule)

Class implementing change point detection algorithms in time series modules.

\_\_dictCost *= None*[¶](#X95e2eb7305ac3945be1bca964e223fd41a620a4)

Dictionary of cost functions, that can be used to in change point detection algorithms.

\_\_outvarCPoints *= None*[¶](#X0bb79f93b6f2bc5fb2a3d8b6c174b6caf25aceb)

How many change points should be found in algorithm.

\_\_outvarCPointsConsider *= None*[¶](#X555a9301dd68c7b4eeb01e63369859ffd07a2be)

If number of change points is known, number of change points is taken as an argument instead of penalty value in change point detection algorithms.

\_\_outvarCost *= None*[¶](#Xb8dbb844089fd267ac14c2e433a0465e3961a53)

Chosen cost function, that can be used to in change point detection algorithms.

\_\_outvarPenalty *= None*[¶](#X6bec8149bf4407849e51b53522ac5a5a007d861)

Linear penalty parameter. Value determines how many change points might detection algorithms find.

buildConfig(*section: tkinter.ttk.Frame*)[¶](#Xf8827bbfa1bbe2652fedcdcfe3c8e6c7cf5b55d)

Provides GUI elements for time series module configuration.

Parameters

**section** – GUI component where module configuration should be displayed.

buildConfigAlgoParams(*section: tkinter.ttk.Frame*)[¶](#X09c859a2b1badae65149b9f68b8cc3edef2f8ee)

Method meant to be overwritten. Provides GUI elements for specific change point detection algorithms’ parameters.

Parameters

**section** – GUI component where specific parameters should be displayed.

calculateChangePoints(*signal*) → list[¶](#X8fba9c6495b02c421d532071935ae8977105249)

Method meant to be overwritten. Calculates change point indexes.

Parameters

**signal** – Time series data.

Returns

Indexes of change points.

displayModule(*ax*, *plotdf: pandas.core.frame.DataFrame*)[¶](#X043a69dcf35ea3b84f6c0c6d2a581bd96dafc20)

Processes analyzing module and displays results on chart’s axes.

Parameters

* **ax** – Reference to plot’s axes.
* **plotdf** – Data frame with original signal.

getCPointsCount() → int[¶](#X029d3af65460cba20baadc550ec2fd69bea5de3)

Returns how many change points should be found in algorithm.

Returns

How many change points should be found in algorithm.

getCostFunction() → str[¶](#X7d1ce84109c9d0e47e63e74e49c3aea764f9d5f)

Returns chosen cost function, that can be used to in change point detection algorithms.

Returns

Chosen cost function, that can be used to in change point detection algorithms.

getLinearPenalty() → float[¶](#Xf6f316abbcc7062c02ddbd276121a55d7bacbf7)

Returns linear penalty parameter. Value determines how many change points might detection algorithms find.

Returns

Linear penalty parameter. Value determines how many change points might detection algorithms find.

shouldConsiderCPoints() → bool[¶](#X93d8085aa884e2c07cfd521b53500d6b1124bfe)

Returns the indicator if number of change should be taken as an argument instead of penalty value in change point detection algorithms.

Returns

The indicator if number of change should be taken as an argument instead of penalty value in change point detection algorithms.

*class* tsmodules.rupturesmods.RupturesPeltModule(*name*)[¶](#Xd42167af9533ac603dcee6ae8410a2abaeeab72)

Bases: [tsmodules.rupturesmods.RupturesModule](#tsmodules.rupturesmods.RupturesModule)

Class implementing Pelt algorithm that detects change points.

calculateChangePoints(*signal*) → list[¶](#Xb9c9085e9873090a7db8b71986b6a11c8ac8007)

Calculates change point indexes.

Parameters

**signal** – Time series data.

Returns

Indexes of change points.

*class* tsmodules.rupturesmods.RupturesWindowModule(*name*)[¶](#Xaf5f66b71c082356567a0f290a6e91ef0eb78d0)

Bases: [tsmodules.rupturesmods.RupturesModule](#tsmodules.rupturesmods.RupturesModule)

Class implementing Window algorithm that detects change points.

\_outvarWidth *= None*[¶](#X420cad480fa6879ff0e2e9f0ece7be934f3ddc3)

Length of the window.

buildConfigAlgoParams(*section: tkinter.ttk.Frame*)[¶](#Xb979f5cb106f5a98afa79e0ab5c79cdda85c867)

Provides GUI elements for specific change point detection algorithms’ parameters.

Parameters

**section** – GUI component where specific parameters should be displayed.

calculateChangePoints(*signal*) → list[¶](#Xf7760c6a3a08427f4ceeba88494eef1d2a6c93a)

Calculates change point indexes.

Parameters

**signal** – Time series data.

Returns

Indexes of change points.

#### Module contents[¶](#module-tsmodules)

# Indices and tables[¶](#indices-and-tables)

* [Index](genindex.html)
* [Module Index](py-modindex.html)
* [Search Page](search.html)

# [timeseries](#Xa39a3ee5e6b4b0d3255bfef95601890afd80709)

### Navigation

Contents:

* [Projekt](index.html#document-modules)

### Related Topics

* [Documentation overview](#Xa39a3ee5e6b4b0d3255bfef95601890afd80709)

©2022, bp. | Powered by [Sphinx 4.3.2](http://sphinx-doc.org/) & [Alabaster 0.7.12](https://github.com/bitprophet/alabaster)