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

ax *= None*[¶](#chart.Chart.ax)

df*: pandas.core.frame.DataFrame = None*[¶](#chart.Chart.df)

display*: tkinter.ttk.Frame = None*[¶](#chart.Chart.display)

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

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

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

id*: int = 0*[¶](#chart.Chart.id)

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

### 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.

buildSections()[¶](#gui.AnalyzerGui.buildSections)

Constructs all sections of an application.

buildSubsectionDataSelection()[¶](#X81780f8f1b8b7287e67cbc191e727ba48add467)

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

buildSubsectionExportData()[¶](#X077dcb7be82adaa8639d1b520d3715d5a35417f)

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

buildSubsectionLoadData()[¶](#gui.AnalyzerGui.buildSubsectionLoadData)

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()[¶](#X4832680b103d35ed54523e63fa55b6ff38ad6c2)

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

buildSubsectionTSModules()[¶](#gui.AnalyzerGui.buildSubsectionTSModules)

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).

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.

statisticstabRefresh()[¶](#gui.AnalyzerGui.statisticstabRefresh)

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

subsectionDataSelection*: tkinter.Frame = None*[¶](#gui.AnalyzerGui.subsectionDataSelection)

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

subsectionDataSelection\_btndelete\_action()[¶](#X3aeaa732144b819d322cbe38b0baf9c94a03c66)

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\_btnupdate\_action()[¶](#Xeccbf52575d97ba0e8509368397fbe91973c798)

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*)[¶](#X28b6c530390b600d661e403c0c4858032a8dcbb)

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\_btnexportchart\_action()[¶](#Xa1a290b949ffc95d149197d643fc1ee1aadd32c)

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\_btnexportlogs\_action()[¶](#X384040892f23aae0b402c4a3f5476ba11fddbd4)

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

subsectionExportData\_btnexportstats\_action()[¶](#X880654af0c4f6244ff2c28305c2c66c0276ca27)

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\_btnexport\_action()[¶](#Xb819d77f19d9c47cdbae213fde6398311ab8620)

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\_btnimport\_action()[¶](#X46f709a2d1e091143f978cedf0ca6b8a604db3d)

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\_outputModulescb\_action(*event*)[¶](#Xd477f70ad79a1e61e7472736ee20d407becc5ad)

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\_tsmodulesentrycb\_action(*event*)[¶](#X8650d0f14b264b42e0f1dad66b77cb7d9b1bf02)

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.

updateChartsCb()[¶](#gui.AnalyzerGui.updateChartsCb)

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

*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.

buildSections()[¶](#gui.GeneratorGui.buildSections)

Constructs all sections of an application.

buildSubSectionChartGenerator()[¶](#X6779bc289656a1e0ba3673f02853a6244fa2738)

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

buildSubSectionExportData()[¶](#X8efcdb59a5a782ad700772f986409f91c85c632)

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

cpCount *= 2*[¶](#gui.GeneratorGui.cpCount)

Number of change points in generated signal.

cps *= None*[¶](#gui.GeneratorGui.cps)

Generated change point indexes.

cpspreadVar *= None*[¶](#gui.GeneratorGui.cpspreadVar)

Variance in areas between change points in generated signal.

df *= Empty DataFrame Columns: [] Index: []*[¶](#gui.GeneratorGui.df)

Data frame of generated signal.

meanVar *= None*[¶](#gui.GeneratorGui.meanVar)

Mean of all data in generated signal.

means *= None*[¶](#gui.GeneratorGui.means)

Means in area between change points, before applying noise.

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.

spreadVar *= None*[¶](#gui.GeneratorGui.spreadVar)

Spread of mean of all data in generated signal.

spreadcpspreadVar *= None*[¶](#gui.GeneratorGui.spreadcpspreadVar)

Spread of variance in areas between change points in generated signal.

spreads *= None*[¶](#gui.GeneratorGui.spreads)

Variances in area between change points, before applying noise.

subsectionChartgenerator*: tkinter.Frame = None*[¶](#Xa84cb5f82fe064f1efb8e5f01445f69302b5980)

Subsection of settings section that configures parameters of generated chart.

subsectionExportdata*: tkinter.Frame = None*[¶](#gui.GeneratorGui.subsectionExportdata)

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

subsectionExportdata\_btnexport\_action()[¶](#X167db03065a35964e2b870194abfebe8eaba1e9)

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\_btnupdate\_action()[¶](#X5355626690c50b69ef8fd83e4e6a49557daa2d4)

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.

validate\_floatentries(*val*)[¶](#gui.GeneratorGui.validate_floatentries)

Checks if value in some text entries is positive float value.

Parameters

**val** – Value to check.

Returns

If val is float positive. Rollbacks operation if false.

*class* gui.Gui[¶](#gui.Gui)

Bases: object

Base class responsible for constructing window application.

buildGui()[¶](#gui.Gui.buildGui)

Constructs and runs root window for an application.

buildSections()[¶](#gui.Gui.buildSections)

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

root *= None*[¶](#gui.Gui.root)

Reference to root window.

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

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

Bases: object

Class with methods concerning loaded data in form of charts.

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.

allCharts *= []*[¶](#hubCharts.HubCharts.allCharts)

List containing loaded data.

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.

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.

id *= 0*[¶](#hubCharts.HubCharts.id)

ID that will be assigned to next generated chart.

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.

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.

allDrivers *= ()*[¶](#hubImporters.HubImporters.allDrivers)

List containing all import methods.

allMethods *= ()*[¶](#hubImporters.HubImporters.allMethods)

List containing all import method names.

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.

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.

allModules *= []*[¶](#hubTSModules.HubTSModules.allModules)

List containing all loaded time series modules.

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.

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*, *displayname*, *defaultTitle*)[¶](#importers.baseimport.ImportMethod)

Bases: object

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

defaultTitle*: str = ''*[¶](#X9cdabcee2867e98b4cadeebd2f44ac62c8a8042)

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

directory*: str = ''*[¶](#Xfddbc3ef3f4f366f06bf506dbb187c10fae5dfd)

Directory where data is located.

displayname*: str = ''*[¶](#X200c4ad49524a247d830307126718ceac6fc1fd)

Format of import method displayed in file dialog.

extension*: str = ''*[¶](#X4186a6a94d9d40e6717df8fe27d70c19f9ce464)

Supported format of import method.

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

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

Returns

Data frame of single, selected time series.

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

Method meant to be overwritten. Provides GUI elements to select specific signal from provided data.

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

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

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

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

datax *= []*[¶](#X48b70112546dd6a6f59b077d419bd47be70bbcc)

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

datay *= []*[¶](#Xa02d0b1f0eec0ab6fd147e5e9490b990996d1cb)

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

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 to select 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.

xseries *= None*[¶](#X05714bfb37c333079f4679b677fa4554c05176f)

Chosen X-axis name from chosen sheet

xseriesprovided *= False*[¶](#X4fb279eea210668411622cc9f0b6bf1b0a001a1)

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

yseries *= None*[¶](#X3e4ae3acafaab6a1fb5011aa0ab7166a8d5a789)

Chosen Y-axis name from chosen sheet.

yseriesprovided *= False*[¶](#Xccda72090a01d916f60d8e6ad8ee933c5be6233)

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

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

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

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

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

datax *= []*[¶](#Xf12b1ef76bc3c34dbbd6550472876c221e333ad)

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

datay *= []*[¶](#Xa8547755771fa7c7a79a09291cad0d6c16223d7)

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

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 to select 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.

sheet *= None*[¶](#Xd04a8674252a3482c9d181949f859027bb6d11c)

Chosen sheet name from spreadsheet.

xseries *= None*[¶](#X27fbca6371edfd074c68d6843b444c7906754ab)

Chosen X-axis name from chosen sheet

xseriesprovided *= False*[¶](#Xdcb56b32963f7da246f2f4a7e9dddaf1fa92613)

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

yseries *= None*[¶](#Xe0b7f1c2682bcd30a1e6cf7682c532ea3cefa77)

Chosen Y-axis name from chosen sheet.

yseriesprovided *= False*[¶](#X3abaaac4fa91333f2d817c2d12c223cb45ad075)

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

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

### loader module[¶](#module-loader)

*class* loader.Loader[¶](#loader.Loader)

Bases: object

*static* loadConfig()[¶](#loader.Loader.loadConfig)

### 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.

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

#### Submodules[¶](#submodules)

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

*class* tsmodules.autocorrelation.AutocorrelationModule(*name*)[¶](#X927fd7bc979f93044250a09cb76e31a4b10f998)

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

buildConfig(*subsection: tkinter.ttk.Frame*)[¶](#X1355fc382d40565850e622b3f8cdf29523a2bc2)

clearOutputDataframe()[¶](#X029e7abb39995d9cb7b1525ce39cf1d0b9fe09d)

displayModule(*ax*, *plotdf*)[¶](#X973447d7e0ebbe0dcd030783225ab38311dc80c)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#X63279bba5c17d4842feb0b3247da8e86c3608bc)

lagentryvar *= None*[¶](#X7fd69f9d4284bf4787def9b922c7d45e4340937)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#X9ccd73b1ef2d2a8b3babf45a83ca3a44f77370b)

validate\_lagentry(*val*)[¶](#Xa6576353296f0618a05a5315f17fb517bebc7ca)

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

*class* tsmodules.basemod.TSModule(*name*)[¶](#tsmodules.basemod.TSModule)

Bases: object

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

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

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

clearOutputDataframe()[¶](#Xd6e109c996fd75b9d06f8187f522acfc1519954)

displayModule(*ax*, *plotdf*)[¶](#tsmodules.basemod.TSModule.displayModule)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#Xf25cd76fb62f669ccb125cdd112498e78f336da)

isactive *= None*[¶](#tsmodules.basemod.TSModule.isactive)

name *= 'Unnamed Module'*[¶](#tsmodules.basemod.TSModule.name)

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

*class* tsmodules.differentation.DifferentiationModule(*name*)[¶](#X672b172bcab45b08cd28bee7ac25fdbf96179a2)

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

clearOutputDataframe()[¶](#X17c1541cfa6874d9063d43c7cfa6dcf4e093d0c)

displayModule(*ax*, *plotdf*)[¶](#Xc3c51b31dedae6f81924640370f392eb8ea89e3)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#Xe5da3c79cec278a5f537d23adfb14038eba4038)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#X0a735f62ede10ccf4b1fa64456df3ffba537d07)

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

*class* tsmodules.originaldata.OriginalDataDisplayModule(*name*)[¶](#Xc8af7577ff4873d4734365c097eda15e965bbf8)

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

clearOutputDataframe()[¶](#X45224d5ce3afc72cfef992c365cdc38c6e43bdd)

displayModule(*ax*, *plotdf*)[¶](#Xd37f55c06157d97f7a75d915af7aee55df3e0bb)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#Xaca8b24aff935ed4d7e19c2e110bc8908f62c01)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#Xcf6ef692bc556e6bd4569f60ae2d9b1025b2dcf)

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

*class* tsmodules.rollingmean.RollingMeanModule(*name*)[¶](#tsmodules.rollingmean.RollingMeanModule)

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

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

clearOutputDataframe()[¶](#X74c3a37d2619f1c51eb004f75d083b02ad73b42)

displayModule(*ax*, *plotdf*)[¶](#X6ddce112c58cd1d9832c5205a18001794933f98)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#X34a32aee32d13ab6eee0d5609087e23f40bc94c)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#X2b94d59cfbf2080a3108a8b63e1d9ec64f8bb0a)

rollCount *= 5*[¶](#Xf424e4249bd405a869631acac130551d9480663)

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

*class* tsmodules.rollingstd.RollingStdModule(*name*)[¶](#tsmodules.rollingstd.RollingStdModule)

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

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

clearOutputDataframe()[¶](#Xa29501f10343d2beb0f4918ade173d950403a03)

displayModule(*ax*, *plotdf*)[¶](#Xde210be89f2793ea039c727c55f794112a42270)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#X422dae39ad28dd00777d70decc67608d2db4cca)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#X9a079b320c102d4a1ea21dc88c4fe519b575047)

rollCount *= 5*[¶](#Xadccb13b717ce8dadb792fadf321c824618e56b)

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

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

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

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

*class* tsmodules.rupturesmods.RupturesDynpModule(*name*)[¶](#Xa25a9bab975f9a604e68cd9cd9a837c23a6720f)

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

calculateChangePoints(*points*) → list[¶](#Xdcee7344561b3a9c5500152e03a83d49879f51b)

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

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

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

clearOutputDataframe()[¶](#X5de159318df5fa4d0e758df0042736f46f68b2d)

displayModule(*ax*, *plotdf*)[¶](#X043a69dcf35ea3b84f6c0c6d2a581bd96dafc20)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#X9b572e80dc06ac6d919eb6adb2d553df5849cfb)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#Xe0475cae103f6c097957be58dc732d08c7b8331)

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

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

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

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

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

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

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

*class* tsmodules.seasonaldecompose.SeasonalDecomposeModule(*name*)[¶](#Xf5cef14745b9515b0f9c68faac82defa5dcac64)

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

buildConfig(*subsection: tkinter.ttk.Frame*)[¶](#Xa1b264730b800f4d862447b120227091943f960)

clearOutputDataframe()[¶](#Xa67f168cebb84d7b5a86a6388efa306174e124e)

componentslist *= ['lag', 'trend', 'resid']*[¶](#X3cdfa39e601d72236b3299a8dd371eba1825bc0)

componentsvarlist *= []*[¶](#X4e9120314e3e9feba5d3ddc918414fbcacdf6c1)

displayModule(*ax*, *plotdf*)[¶](#X8151cf5c4f9657ac762457e0acd7486ae9b155b)

getOutputDataframe() → pandas.core.frame.DataFrame[¶](#X1d9bccc6cb89392eba616c6d16db0b83cb08e51)

lagentryvar *= None*[¶](#X2bfca80d4091f9dcc6033811112072cfa476336)

outputDataframe *= Empty DataFrame Columns: [] Index: []*[¶](#X12d8ed4be75f141ea76c0178512ab0f6bdfa458)

validate\_lagentry(*val*)[¶](#X432b80e703c6f2c44296dbb1597fa4be4255eb1)

#### 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)