
TimeSeriesX Documentation

Release 0.1.13

Alexander Schulz

Jul 29, 2022

CONTENTS:

1	TimeSeriesX	1
1.1	About	1
1.2	Features	1
1.3	Credits	2
2	Installation	3
2.1	Stable release	3
2.2	From sources	3
3	Usage	5
4	timeseriesx	7
4.1	timeseriesx package	7
5	Contributing	17
5.1	Types of Contributions	17
5.2	Get Started!	18
5.3	Pull Request Guidelines	19
5.4	Tips	19
5.5	Deploying	19
6	Credits	21
6.1	Development Lead	21
6.2	Contributors	21
7	History	23
7.1	0.1.13 (2022-07-19)	23
7.2	0.1.12 (2022-03-16)	23
7.3	0.1.11 (2022-03-07)	23
7.4	0.1.10 (2022-01-21)	23
7.5	0.1.9 (2021-11-19)	24
7.6	0.1.8 (2021-09-28)	24
7.7	0.1.7 (2021-09-28)	24
7.8	0.1.6 (2021-09-13)	24
7.9	0.1.5 (2021-09-10)	24
7.10	0.1.4 (2021-09-09)	25
7.11	0.1.3 (2021-09-08)	25
7.12	0.1.2 (2021-09-07)	25
7.13	0.1.1 (2021-02-16)	25
8	Indices and tables	27

Python Module Index	29
Index	31

TIMESERIESX

The eXtended time series library.

Manage time series data with explicit time zone, frequency and unit.

- Free software: MIT license
- Documentation: <https://timeseriesx.readthedocs.io>.

1.1 About

TimeSeriesX is motivated by handling time series data in a convenient way. Almost all the features are actually already provided by `pandas`. TimeSeriesX extends the `pandas` time series functionality by the unit functionalities of `pint` and `pint-pandas`. Further, TimeSeriesX offers an easy and convenient interface to work with time series without the need to dig deep into these libraries, which nevertheless is still recommended, since they go way beyond time series data.

The main challenges that arise when handling time series data are time zones and frequencies. Since time series data is often obtained by measurements, the values are associated with units. Then these units can be confused easily, since the units are often not modeled in code.

TimeSeriesX forces the user to handle time zones, frequencies and units explicitly, while taking care of validation and convenient formats. It also supports deriving these attributes from raw time series data. It offers a limited set of actions on time series that are translated to `pandas` or `pint` functionality under the hood. It was designed to guarantee that every transformation of time series data results in a new valid time series, which would require quite some `pandas` code if done “manually”.

1.2 Features

- model time series data with explicit frequency, time zone and unit
- convert time zone or unit
- resample data to new frequency
- fill and get gaps
- join time series

- perform calculations on time series with python standard operators

1.3 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

INSTALLATION

2.1 Stable release

To install TimeSeriesX, run this command in your terminal:

```
$ pip install timeseriesx
```

This is the preferred method to install TimeSeriesX, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

2.2 From sources

The sources for TimeSeriesX can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/carlculator/timeseriesx
```

Or download the [tarball](#):

```
$ curl -OJL https://github.com/carlculator/timeseriesx/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```


To use TimeSeriesX in a project:

```
import timeseriesx as tx
```

Create a new series:

```
timestamps = [  
    dt.datetime(2020, 3, 1, 15, 0, 0),  
    dt.datetime(2020, 3, 1, 16, 0, 0),  
    dt.datetime(2020, 3, 1, 17, 0, 0),  
]  
values = [0., 1., 2.]  
ts = tx.TimestampSeries.create_from_lists(timestamps, values, freq='H', unit='km')
```

Convert time zone:

```
# set time zone info  
ts = ts.convert_time_zone('Europe/Stockholm')  
  
# convert time zone  
ts = ts.convert_time_zone('UTC')
```

Convert unit:

```
ts = ts.convert_unit('nautical_mile')
```

Resample frequency:

```
ts = ts.resample(  
    '1D', # one day  
    method='sum'  
)
```

Perform calculations:

```
ts = ts + [2., 3., 4]  
ts = ts * ts  
ts = ts / 42.
```


TIMESERIESX

4.1 timeseriesx package

4.1.1 Subpackages

timeseriesx.base package

Submodules

timeseriesx.base.base_time_series module

class timeseriesx.base.base_time_series.BaseTimeSeries

Bases: object

aggregate(*func*)

aggregate all values of the series with a custom aggregation function

Parameters *func* (*function*) – a function mapping a numeric list/array/vector to a scalar

Returns the aggregated value

Return type numpy.float/numpy.int

append(*value*)

abstract as_dict(*ordered=False*)

as_pd_series(*include_nan=True*)

abstract as_tuples()

property empty

property first

join(*other_ts*, *fit=True*)

property last

map(*func*, ***kwargs*)

apply a custom function to each value of the series

Parameters *func* (*function*) – a function mapping a scalar to another scalar

Returns self

Return type *BaseTimeSeries*

`mean()`
`prepend(value)`
`round(decimals)`
`sum()`
property values

timeseriesx.base.timestamp_series module

exception `timeseriesx.base.timestamp_series.TimestampMismatchWarning`

Bases: `RuntimeWarning`

warning about implicit handling of mismatching timestamps

class `timeseriesx.base.timestamp_series.TimestampSeries(series, freq=None, unit=None, time_zone=None)`

Bases: `timeseriesx.mixins.unit.UnitMixin`, `timeseriesx.mixins.time_zone.TimeZoneMixin`,
`timeseriesx.mixins.frequency.FrequencyMixin`, `timeseriesx.base.base_time_series.BaseTimeSeries`

append(*value*)

append a new value to a series with frequency

Parameters *value* (*float/int*) – the value to append

Returns the series with the new appended value

Return type *TimestampSeries*

as_dict(*ordered=False*)

as_time_period_series(*align_left=True*)

as_tuples()

static **create_constant_timeseries**(*start, end, value, freq, unit=None, time_zone='infer'*)

create a *TimestampSeries*-object from *start* to *end* with constant value

Parameters

- **start** (*str/datetime.datetime/pandas.Timestamp*) – the start timestamp of the series (included)
- **end** (*str/datetime.datetime/pandas.Timestamp*) – the end timestamp of the series (included)
- **value** (*int/float*) – the constant value for each element
- **freq** (*str/datetime.timedelta/pandas.Offset/pandas.Timedelta*) – the frequency of the timestamp series, *pandas offset aliases* supported
- **unit** (*str/pint.Unit*) – the unit of the series's values, many string representations of common units are supported, such as *m*, *s*, *kg* and many more
- **time_zone** (*str/tzinfo*) – the name of the time zone, (see [IANA](#)) or a *tzinfo*-object, pass *'infer'* if you want the time zone to be derived from *start* and *end*

Returns a new *TimestampSeries*-object

Return type *TimestampSeries*

static create_from_dict(*dictionary*, *freq*='infer', *unit*=None, *time_zone*='infer')
 create a *TimestampSeries*-object from a dict timestamps as keys and values as values

Parameters

- **dictionary** (*dict*) – dict with timestamps as keys and timeseries-values as dict-values
- **freq** (*str*/*datetime.timedelta*/*pandas.Offset*/*pandas.Timedelta*) – the frequency of the timestamp series, [pandas offset aliases](#) supported, pass 'infer' if you want the frequency to be derived by the timestamps
- **unit** (*str*/*pint.Unit*) – the unit of the series's values, many string representations of common units are supported, such as *m*, *s*, *kg* and many more
- **time_zone** (*str*/*tzinfo*) – the name of the time zone, (see [IANA](#)) or a *tzinfo*-object, pass 'infer' if you want the time zone to be derived by the timestamps

Returns a new *TimestampSeries*-object

Return type *TimestampSeries*

static create_from_lists(*timestamps*, *values*, *freq*='infer', *unit*=None, *time_zone*='infer')
 create a *TimestampSeries*-object from a list of timestamps and values matched by their index

Parameters

- **timestamps** (*list*) – the timestamps of the series
- **values** (*list*) – the values of the series
- **freq** (*str*/*datetime.timedelta*/*pandas.Offset*/*pandas.Timedelta*) – the frequency of the timestamp series, [pandas offset aliases](#) supported, pass 'infer' if you want the frequency to be derived by the timestamps
- **unit** (*str*/*pint.Unit*) – the unit of the series's values, many string representations of common units are supported, such as *m*, *s*, *kg* and many more
- **time_zone** (*str*/*tzinfo*) – the name of the time zone, (see [IANA](#)) or a *tzinfo*-object, pass 'infer' if you want the time zone to be derived by the timestamps

Returns a new *TimestampSeries*-object

Return type *TimestampSeries*

static create_from_pd_series(*series*, *freq*='infer', *unit*=None, *time_zone*='infer')
 create a *TimestampSeries*-object from a pandas *Series* with *DatetimeIndex*

Parameters

- **pandas.Series** – a pandas series-object with *DatetimeIndex*
- **freq** (*str*/*datetime.timedelta*/*pandas.Offset*/*pandas.Timedelta*) – the frequency of the timestamp series, [pandas offset aliases](#) supported, pass 'infer' if you want the frequency to be derived by the timestamps
- **unit** (*str*/*pint.Unit*) – the unit of the series's values, many string representations of common units are supported, such as *m*, *s*, *kg* and many more
- **time_zone** (*str*/*tzinfo*) – the name of the time zone, (see [IANA](#)) or a *tzinfo*-object, pass 'infer' if you want the time zone to be derived from *start* and *end*

Returns a new *TimestampSeries*-object

Return type *TimestampSeries*

static create_from_tuples(*tuples*, *freq*='infer', *unit*=None, *time_zone*='infer')
create a *TimestampSeries*-object from a list of tuples of timestamps and values

Parameters

- **tuples** (*list*) – list of tuples holding timestamp and value
- **freq** (*str/datetime.timedelta/pandas.Offset/pandas.Timedelta*) – the frequency of the timestamp series, [pandas offset aliases](#) supported, pass 'infer' if you want the frequency to be derived by the timestamps
- **unit** (*str/pint.Unit*) – the unit of the series's values, many string representations of common units are supported, such as *m*, *s*, *kg* and many more
- **time_zone** (*str/tzinfo*) – the name of the time zone, (see [IANA](#)) or a *tzinfo*-object, pass 'infer' if you want the time zone to be derived by the timestamps

Returns a new *TimestampSeries*-object

Return type *TimestampSeries*

static create_null_timeseries(*start*, *end*, *freq*, *unit*=None, *time_zone*='infer')
create a *TimestampSeries*-object from *start* to *end* with NaN-values

Parameters

- **start** (*str/datetime.datetime/pandas.Timestamp*) – the start timestamp of the series (included)
- **end** (*str/datetime.datetime/pandas.Timestamp*) – the end timestamp of the series (included)
- **freq** (*str/datetime.timedelta/pandas.Offset/pandas.Timedelta*) – the frequency of the timestamp series, [pandas offset aliases](#) supported
- **unit** (*str/pint.Unit*) – the unit of the series's values, many string representations of common units are supported, such as *m*, *s*, *kg* and many more
- **time_zone** (*str/tzinfo*) – the name of the time zone, (see [IANA](#)) or a *tzinfo*-object, pass 'infer' if you want the time zone to be derived from *start* and *end*

Returns a new *TimestampSeries*-object

Return type *TimestampSeries*

property end

property first

join(*other_ts*, *fit*=True)

property last

map(*func*, *dimensionless*=True)

apply a custom function to each value of the series

Parameters

- **func** (*function*) – a function mapping a scalar to another scalar
- **dimensionless** (*bool*) – if set to True, the mapping function takes an argument of type *Number* (no unit, dimensionless). The resulting timestamp series will keep the original unit. If set to False, the mapping function takes an argument of type *pint.Quantity*. The resulting timestamp series will have the unit of the mapped values. Mapping values of one series to different units results in an error. Mapping with *dimensionless*=False will result in a loop and therefore perform slower.

Returns the series with mapped values

Return type *TimestampSeries*

prepend(*value*)

prepend a new value to a series with frequency

Parameters **value** (*float/int*) – the value to prepend

Returns the series with the new prepended value

Return type *TimestampSeries*

round(*decimals*)

round the values of the series

Parameters **decimals** – no of decimal places to round to

Returns the series with rounded values

Return type *TimestampSeries*

property **start**

property **time_range**

property **timestamps**

validate_all()

property **values**

Module contents

timeseriesx.mixins package

Submodules

timeseriesx.mixins.frequency module

class timeseriesx.mixins.frequency.**FrequencyMixin**(*args, **kwargs)

Bases: *timeseriesx.mixins.BaseMixin*

fill_gaps(*start=None, end=None, value=nan*)

fill all gaps between *start* and *end* in a series with a frequency with a constant value

Parameters

- **start** (*datetime.datetime*) – the start timestamps of the period that will be investigated (included). If *None*, then the first timestamp in the time series is considered as start. Defaults to *None*
- **end** (*datetime.datetime*) – the end timestamps of the period that will be investigated (included). If *None*, then the last timestamp in the time series is considered as end. Defaults to *None*
- **value** (*float/int/np.float*) – the constant fill value

Returns return the series with filled gaps

Return type *BaseTimeSeries*

property **freq**

get_gaps(*start=None, end=None*)

get all timestamps between *start* and *end* from a series with a frequency, where the value is missing or NaN

Parameters

- **start** (*datetime.datetime*) – the start timestamps of the period that will be investigated (included). If *None*, then the first timestamp in the time series is considered as start. Defaults to *None*
- **end** (*datetime.datetime*) – the end timestamps of the period that will be investigated (included). If *None*, then the last timestamp in the time series is considered as end. Defaults to *None*

Returns list of timestamps

Return type list of *datetime.datetime*

resample(*freq, method*)

resample the series to a smaller frequency, aggregate the values

Parameters

- **freq** (*str/datetime.timedelta/pandas.Offset/pandas.Timedelta*) – the new frequency, has to be smaller than the current frequency (greater offset)
- **method** (*str/Callable*) – aggregation method, currently supported are “all”, “any”, “min”, “max”, “sum”, “mean”, “median”, or function that a collection (e.g. *pandas.Series* or list) of numeric values as its argument and returns a scalar

Returns the resamples time series

Return type *BaseTimeSeries*

timeseriesx.mixins.time_zone module

class *timeseriesx.mixins.time_zone.TimeZoneMixin*(*args, **kwargs)

Bases: *timeseriesx.mixins.BaseMixin*

convert_time_zone(*tz*)

convert time series index to another time zone, or make an time zone naive index time zone aware (or the other way round)

Parameters **tz** (*str/datetime.tzinfo*) – *tzinfo* object or name of the new time zone or *None*

Returns the series with converted index

Return type *BaseTimeSeries*

property *time_zone*

exception *timeseriesx.mixins.time_zone.TimeZoneWarning*

Bases: *RuntimeWarning*

warning about implicit time zone handling

timeseriesx.mixins.unit module

class timeseriesx.mixins.unit.**UnitMixin**(*args, **kwargs)

Bases: [timeseriesx.mixins.BaseMixin](#)

aggregate(func, with_unit=False)

aggregate all values of the series with a custom aggregation function

Parameters

- **func** (*function*) – a function mapping a numeric list/array/vector to a scalar
- **with_unit** (*boolean*) – flag whether to return the result as a pint object, defaults to False

Returns the aggregated value

Return type numpy.float/numpy.int/pint.Quantity

as_pd_series(include_nan=True)

convert_unit(unit)

convert the unit of the series

Parameters **unit** (*str/pint.Unit*) –

Returns the time series with converted units

Return type [BaseTimeSeries](#)

mean(with_unit=False)

calculate the mean of all values of the series

Parameters **with_unit** (*boolean*) – flag whether to return the result as a pint object, defaults to False

Returns the mean of the values

Return type numpy.float/numpy.int/pint.Quantity

sum(with_unit=False)

calculate the sum of all values of the series

Parameters **with_unit** (*boolean*) – flag whether to return the result as a pint object, defaults to False

Returns the sum of the values

Return type numpy.float/numpy.int/pint.Quantity

property unit

exception timeseriesx.mixins.unit.**UnitWarning**

Bases: RuntimeWarning

warning about implicit unit handling

Module contents

`class timeseriesx.mixins.BaseMixin(*args, **kwargs)`
Bases: object

timeseriesx.validation package

Submodules

timeseriesx.validation.frequency module

`timeseriesx.validation.frequency.coerce_freq(freq)`
return a convenient representation of a frequency as a `pandas.DateOffset` object

Parameters `freq` (`str/datetime.timedelta/pandas.Timedelta/pandas.DateOffset`) – a frequency string or frequency object to coerce

Returns coerced frequency object

Return type `pandas.DateOffset`

`timeseriesx.validation.frequency.infer_freq(series)`
infer the frequency from a `pandas.Series` with `DatetimeIndex`

Parameters `series` (`pandas.Series`) – a `pandas.Series` with `DatetimeIndex`

Returns the inferred frequency object

Return type `pandas.DateOffset`

timeseriesx.validation.time_zone module

`timeseriesx.validation.time_zone.coerce_time_zone(tz)`
returns the convenient representation of a time zone as a pytz-based `tzinfo`-object

Parameters `tz` (`str/datetime.tzinfo`) –

Returns the coerced time zone object

Return type `datetime.tzinfo`

`timeseriesx.validation.time_zone.infer_tz_from_series(series)`
infer the the time zone from a `pandas series` with `DatetimeIndex`

Parameters `series` (`pandas.Series`) – the target series

Returns the inferred time zone

Return type `datetime.tzinfo`

`timeseriesx.validation.time_zone.infer_tz_from_timestamp(timestamp)`
infer the the time zone from a timestamp object

Parameters `timestamp` (`pandas.Timestamp/datetime.datetime`) – the target timestamp

Returns the inferred time zone

Return type `datetime.tzinfo`

timeseriesx.validation.timestamp_index module

`timeseriesx.validation.timestamp_index.index_is_datetime(series, exception=True)`
check if the index of a pandas.Series is a valid DatetimeIndex

Parameters

- **series** (*pandas.Series*) – the series holding the index to check
- **exception** (*bool*) – if True, raise an exception in case of invalid DatetimeIndex

Returns True if index is a valid DatetimeIndex, False otherwise.

Return type bool

`timeseriesx.validation.timestamp_index.index_is_sorted(series, ascending=True, exception=True)`
check if the (datetime-) index of a pandas.Series is sorted

Parameters

- **series** (*pandas.Series*) – the series holding the index to check
- **ascending** (*bool*) – if true, check for ascending order, if false for descending order
- **exception** (*bool*) – if True, raise an exception in case of unsorted index

Returns True if index is sorted, False otherwise.

Return type bool

timeseriesx.validation.unit module

`timeseriesx.validation.unit.coerce_unit(unit)`
returns the convenient representation of a unit as a *pint.Unit*-object

Parameters **unit** (*str/pint.Unit*) – the unit string to parse or a Unit object

Returns the coerced unit

Return type *pint.Unit*

Module contents

4.1.2 Module contents

Top-level package for TimeSeriesX.

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at <https://github.com/carlculator/timeseriesx/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

5.1.4 Write Documentation

TimeSeriesX could always use more documentation, whether as part of the official TimeSeriesX docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/carculator/timeseriesx/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *timeseriesx* for local development.

1. Fork the *timeseriesx* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/timeseriesx.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv timeseriesx
$ cd timeseriesx/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 timeseriesx tests
$ python setup.py test or pytest
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.7, 3.8 and 3.9, and for PyPy. Check https://travis-ci.com/carlculator/timeseriesx/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_timeseriesx
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CREDITS

6.1 Development Lead

- Alexander Schulz <info@alexander-schulz.eu>

6.2 Contributors

- [saelsa](#)
- [TimoGuenther](#)

HISTORY

7.1 0.1.13 (2022-07-19)

- fix a few bugs by avoiding *is_compatible_with* in *convert_unit*
- raise only *ValueError* instead of *DimensionalityError* on unit dimensionality mismatch
- remove *pandas* installation dependency because of transitive dependency via *pint-pandas*
- loosen requirements for *pytz* and *dateutil*, no special version requirements known
- extend *__getitem__* functionality by supporting iterables of timestamps or positional indices
- explicitly support indexing by time zone naive timestamps, which is deprecated by *pandas*
- make *coerce_unit* behave like *coerce_freq* and *coerce_time_zone* by passing through *None*

7.2 0.1.12 (2022-03-16)

- fix equals method
- update documentation

7.3 0.1.11 (2022-03-07)

- fix resampling method to support nan-values
- update dependencies

7.4 0.1.10 (2022-01-21)

- fix equals method
- update dependencies

7.5 0.1.9 (2021-11-19)

- allow aggregation functions to return magnitudes or quantities
- update dependencies

7.6 0.1.8 (2021-09-28)

- fix time zone bug in gap handling
- update dependencies
- add more tests

7.7 0.1.7 (2021-09-28)

- improve gap handling
- update dependencies
- improve documentation
- fix calculations with quantity scalar

7.8 0.1.6 (2021-09-13)

- fix time zone issue with UTC in basic calculations for TimestampSeries as 2nd operand
- update pint-pandas version dependency
- use pint's default unit registry
- add support of callables as arguments for frequency resampling
- add more tests

7.9 0.1.5 (2021-09-10)

- fix time zone issue with UTC in basic calculations
- add round-method for TimestampSeries
- fix map-function for series with unit
- add more tests

7.10 0.1.4 (2021-09-09)

- improve test coverage
- improve TimeSeries equality check
- support NaN-removal in `as_pd_series`-method

7.11 0.1.3 (2021-09-08)

- remove manual timezone checks because it is handled by pandas
- fix skipped tests
- fix `repr()` method of `TimestampSeries`
- fix basic calculation with units involved

7.12 0.1.2 (2021-09-07)

- fix timezone handling
- First release on PyPI Index.

7.13 0.1.1 (2021-02-16)

- First release on PyPI Test Index.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

t

- [timeseriesx](#), 15
- [timeseriesx.base](#), 11
 - [timeseriesx.base.base_time_series](#), 7
 - [timeseriesx.base.timestamp_series](#), 8
- [timeseriesx.mixins](#), 14
 - [timeseriesx.mixins.frequency](#), 11
 - [timeseriesx.mixins.time_zone](#), 12
 - [timeseriesx.mixins.unit](#), 13
- [timeseriesx.validation](#), 15
 - [timeseriesx.validation.frequency](#), 14
 - [timeseriesx.validation.time_zone](#), 14
 - [timeseriesx.validation.timestamp_index](#), 15
 - [timeseriesx.validation.unit](#), 15

INDEX

A

aggregate() (*timeseriesx.base.base_time_series.BaseTimeSeries* method), 7
 aggregate() (*timeseriesx.mixins.unit.UnitMixin* method), 13
 append() (*timeseriesx.base.base_time_series.BaseTimeSeries* method), 7
 append() (*timeseriesx.base.timestamp_series.TimestampSeries* method), 8
 as_dict() (*timeseriesx.base.base_time_series.BaseTimeSeries* method), 7
 as_dict() (*timeseriesx.base.timestamp_series.TimestampSeries* method), 8
 as_pd_series() (*timeseriesx.base.base_time_series.BaseTimeSeries* method), 7
 as_pd_series() (*timeseriesx.mixins.unit.UnitMixin* method), 13
 as_time_period_series() (*timeseriesx.base.timestamp_series.TimestampSeries* method), 8
 as_tuples() (*timeseriesx.base.base_time_series.BaseTimeSeries* method), 7
 as_tuples() (*timeseriesx.base.timestamp_series.TimestampSeries* method), 8
 convert_unit() (*timeseriesx.mixins.unit.UnitMixin* method), 13
 create_constant_timeseries() (*timeseriesx.base.timestamp_series.TimestampSeries* static method), 8
 create_from_dict() (*timeseriesx.base.timestamp_series.TimestampSeries* static method), 8
 create_from_lists() (*timeseriesx.base.timestamp_series.TimestampSeries* static method), 9
 create_from_pd_series() (*timeseriesx.base.timestamp_series.TimestampSeries* static method), 9
 create_from_tuples() (*timeseriesx.base.timestamp_series.TimestampSeries* static method), 9
 create_null_timeseries() (*timeseriesx.base.timestamp_series.TimestampSeries* static method), 10

E

empty (*timeseriesx.base.base_time_series.BaseTimeSeries* property), 7
 end (*timeseriesx.base.timestamp_series.TimestampSeries* property), 10

B

BaseMixin (class in *timeseriesx.mixins*), 14
 BaseTimeSeries (class in *timeseriesx.base.base_time_series*), 7

C

coerce_freq() (in module *timeseriesx.validation.frequency*), 14
 coerce_time_zone() (in module *timeseriesx.validation.time_zone*), 14
 coerce_unit() (in module *timeseriesx.validation.unit*), 15
 convert_time_zone() (*timeseriesx.mixins.time_zone.TimeZoneMixin* method), 12

F

fill_gaps() (*timeseriesx.mixins.frequency.FrequencyMixin* method), 11
 first (*timeseriesx.base.base_time_series.BaseTimeSeries* property), 7
 first (*timeseriesx.base.timestamp_series.TimestampSeries* property), 10
 freq (*timeseriesx.mixins.frequency.FrequencyMixin* property), 11
 FrequencyMixin (class in *timeseriesx.mixins.frequency*), 11

G

get_gaps() (*timeseriesx.mixins.frequency.FrequencyMixin* method), 11

I

`index_is_datetime()` (in module `time-seriesx.validation.timestamp_index`), 15

`index_is_sorted()` (in module `time-seriesx.validation.timestamp_index`), 15

`infer_freq()` (in module `time-seriesx.validation.frequency`), 14

`infer_tz_from_series()` (in module `time-seriesx.validation.time_zone`), 14

`infer_tz_from_timestamp()` (in module `time-seriesx.validation.time_zone`), 14

J

`join()` (`timeseriesx.base.base_time_series.BaseTimeSeries` method), 7

`join()` (`timeseriesx.base.timestamp_series.TimestampSeries` method), 10

L

`last()` (`timeseriesx.base.base_time_series.BaseTimeSeries` property), 7

`last()` (`timeseriesx.base.timestamp_series.TimestampSeries` property), 10

M

`map()` (`timeseriesx.base.base_time_series.BaseTimeSeries` method), 7

`map()` (`timeseriesx.base.timestamp_series.TimestampSeries` method), 10

`mean()` (`timeseriesx.base.base_time_series.BaseTimeSeries` method), 7

`mean()` (`timeseriesx.mixins.unit.UnitMixin` method), 13

module

- `timeseriesx`, 15
- `timeseriesx.base`, 11
- `timeseriesx.base.base_time_series`, 7
- `timeseriesx.base.timestamp_series`, 8
- `timeseriesx.mixins`, 14
- `timeseriesx.mixins.frequency`, 11
- `timeseriesx.mixins.time_zone`, 12
- `timeseriesx.mixins.unit`, 13
- `timeseriesx.validation`, 15
- `timeseriesx.validation.frequency`, 14
- `timeseriesx.validation.time_zone`, 14
- `timeseriesx.validation.timestamp_index`, 15
- `timeseriesx.validation.unit`, 15

P

`prepend()` (`timeseriesx.base.base_time_series.BaseTimeSeries` method), 8

`prepend()` (`timeseriesx.base.timestamp_series.TimestampSeries` method), 11

R

`resample()` (`timeseriesx.mixins.frequency.FrequencyMixin` method), 12

`round()` (`timeseriesx.base.base_time_series.BaseTimeSeries` method), 8

`round()` (`timeseriesx.base.timestamp_series.TimestampSeries` method), 11

S

`start()` (`timeseriesx.base.timestamp_series.TimestampSeries` property), 11

`sum()` (`timeseriesx.base.base_time_series.BaseTimeSeries` method), 8

`sum()` (`timeseriesx.mixins.unit.UnitMixin` method), 13

T

`time_range()` (`timeseriesx.base.timestamp_series.TimestampSeries` property), 11

`time_zone()` (`timeseriesx.mixins.time_zone.TimeZoneMixin` property), 12

`timeseriesx`

- module, 15
- `timeseriesx.base`
- module, 11
- `timeseriesx.base.base_time_series`
- module, 7
- `timeseriesx.base.timestamp_series`
- module, 8
- `timeseriesx.mixins`
- module, 14
- `timeseriesx.mixins.frequency`
- module, 11
- `timeseriesx.mixins.time_zone`
- module, 12
- `timeseriesx.mixins.unit`
- module, 13
- `timeseriesx.validation`
- module, 15
- `timeseriesx.validation.frequency`
- module, 14
- `timeseriesx.validation.time_zone`
- module, 14
- `timeseriesx.validation.timestamp_index`
- module, 15
- `timeseriesx.validation.unit`
- module, 15
- `TimestampMismatchWarning`, 8
- `timestamps()` (`timeseriesx.base.timestamp_series.TimestampSeries` property), 11
- `TimestampSeries` (class in `time-seriesx.base.timestamp_series`), 8
- `TimeZoneMixin` (class in `timeseriesx.mixins.time_zone`), 12

[TimeZoneWarning](#), 12

U

[unit](#) (*timeseriesx.mixins.unit.UnitMixin* property), 13

[UnitMixin](#) (*class in timeseriesx.mixins.unit*), 13

[UnitWarning](#), 13

V

[validate_all\(\)](#) (*time-seriesx.base.timestamp_series.TimestampSeries* method), 11

[values](#) (*timeseriesx.base.base_time_series.BaseTimeSeries* property), 8

[values](#) (*timeseriesx.base.timestamp_series.TimestampSeries* property), 11