xrview Documentation

Release 0.2.1

Peter Hausamann

Oct 19, 2020

CONTENTS

Visualizing xarray data with bokeh.

The code is hosted on GitHub: https://github.com/phausamann/xrview

CHAPTER

ONE

INSTALLATION

1.1 Latest release

xrview is currently unreleased, but can be installed from the GitHub repository via `pip`:

```
$ pip install git+https://github.com/phausamann/xrview.git
```

This will also install the minimal dependencies for creating HTML plots.

1.2 Optional dependencies

For plotting in a jupyter notebook, you will also need the `notebook` and `requests` packages:

```
$ pip install notebook requests
```

Bokeh server support depends on a specific version of `tornado`:

```
$ pip install tornado<=4.5.3
```

CHAPTER

TWO

USAGE

xrview provides several utilities for automatically creating interactive bokeh plots from xarray data types.

2.1 Basic plotting with HTML output

`xrview.plot` will create a plot of an `xarray.DataArray` or `Dataset` given the name of the dimension that represents the x coordinate in the plot.

All examples in this section assume the following imports:

```
>>> import numpy as np
>>> import xarray as xr
>>> import xrview
```

2.1.1 Minimal example

The following code will open a browser tab with the figure shown below.

```
>>> x = np.linspace(0, 1, 100)
>>> y = np.sqrt(x)
>>> da = xr.DataArray(y, {'x': x}, 'x')
>>> plot = xrview.plot(da, x='x')
>>> plot.show()
```

2.1.2 Overlaying and tiling plots

When passing a `Dataset`, each variable will be plotted in a separate figure. If the variable has a second dimension, each element along this dimension will be plotted as a separate line and a legend will be automatically created based on the coordinates of this dimension.

```
>>> x = np.linspace(0, 1, 100)
>>> y = np.vstack([np.sqrt(x), x, x ** 2]).T
>>> ds = xr.Dataset({'Clean': (['x', 'f'], y),
...                  'Noisy': (['x', 'f'], y + 0.01*np.random.randn(100, 3))},
...                 {'x': x, 'f': ['sqrt(x)', 'x', 'x^2']})
>>> plot = xrview.plot(ds, x='x', ncols=2)
>>> plot.show()
```

Alternatively, you can show the elements of the dimension in separate figures and overlay the variables by specifying `overlay='data_vars'`:

```
>>> plot = xrview.plot(ds, x='x', ncols=2, overlay='data_vars')
>>> plot.show()
```

You can add additional figures to the plot with the `add_figure()` method by providing data as a DataArray. The DataArray has to contain a coordinate with the same name as the x coordinate, but they do not need to have the same values.

```
>>> da = xr.DataArray(np.ones(20), {'x': np.linspace(0, 1, 20)}, 'x')
>>> plot = xrview.plot(ds, x='x', ncols=2)
>>> plot.add_figure(da)
>>> plot.show()
```

Data can also be overlaid onto existing figures with the `add_overlay()` method. With the `onto` parameter, you can select a figure by index or title on which to overlay the data. By default, the data will be overlaid onto all figures.

```
>>> plot = xrview.plot(ds, x='x', ncols=2)
>>> plot.add_overlay(da, onto='Clean')
>>> plot.show()
```

2.1.3 Customizing glyphs and annotations

xrview supports plotting with many of the standard bokeh glyphs as well as some custom composite glyphs such as error bars and boxes for box plots. Glyphs can be passed to *`plot()`*, `add_overlay()` and `add_figure()` as strings, instances of a glyph class or iterables of any combination of both. When you pass a glyph instance, you can specify additional keyword arguments.

In this example, one array is plotted with circles and a second with a blue line and red squares:

```
>>> from xrview.glyphs import Square
>>> x = np.linspace(0, 1, 100)
>>> y = np.sqrt(x)
>>> da_sqrt = xr.DataArray(y, {'x': x}, 'x')
>>> da_const = xr.DataArray(np.ones(20), {'x': x[::5]}, 'x')
>>> plot = xrview.plot(da_sqrt, x='x', glyphs='circle')
>>> plot.add_overlay(da_const, glyphs=['line', Square(color='red')])
>>> plot.show()
```

xrview also provides a straightforward way of

Note: See *Glyphs* for a list of available glyphs.

2.1.4 Categorical and time series data

2.2 Fine-tuning bokeh figures

2.3 Interactive plotting

Note: Interactive plotting is so far only supported in jupyter notebooks.

2.3.1 Adding interactions

2.3.2 Sub-sampled timeseries plots

CHAPTER

THREE

API REFERENCE

3.1 Top-level functions

Module: `xrview`

plot	Create a plot from xarray data.

3.1.1 xrview.plot

`xrview.`**`plot`**(*X*, *output='html'*, *server=False*, ***kwargs*)

Create a plot from xarray data.

Parameters

X: xarray.DataArray or Dataset The data to be plotted.

output: 'html' or 'notebook', default 'html' Whether to show the plot in an HTML file or a Jupyter notebook output cell.

server: bool, default False If True, create a bokeh server app that supports interactions and plotting of large datasets.

kwargs: Keyword arguments to be passed to the plot instance.

Returns

plot: xrview.BasePanel A plot instance depending on the options.

3.2 HTML

Module: `xrview.html`

HtmlPlot	Base class for HTML plots.
HtmlGridPlot	An HTML grid plot.
HtmlSpacer	An HTML spacer.

3.2.1 xrview.html.HtmlPlot

class xrview.html.**HtmlPlot**(*data*, *x*, *overlay='dims'*, *coords=None*, *glyphs='line'*, *title=None*, *share_y=False*, *tooltips=None*, *tools=None*, *toolbar_location='right'*, *figsize=600, 300*, *ncols=1*, *palette=None*, *ignore_index=False*, *theme=None*, ***fig_kwargs*)

Base class for HTML plots.

Examples

```
import numpy as np
import xarray as xr
from xrview.html import HtmlPlot

x = np.linspace(0, 1, 100)
y = np.sqrt(x)
da = xr.DataArray(y, coords={'x': x}, dims='x')

plot = HtmlPlot(da, x='x')
plot.show()
```

__init__(*data*, *x*, *overlay='dims'*, *coords=None*, *glyphs='line'*, *title=None*, *share_y=False*, *tooltips=None*, *tools=None*, *toolbar_location='right'*, *figsize=600, 300*, *ncols=1*, *palette=None*, *ignore_index=False*, *theme=None*, ***fig_kwargs*)

Constructor.

Parameters

data [xarray DataArray or Dataset] The data to display.

x [str] The name of the dimension in `data` that contains the x-axis values.

glyphs [str, BaseGlyph or iterable, default 'line'] The glyph to use for plotting.

figsize [iterable, default (600, 300)] The size of the figure in pixels.

ncols [int, default 1] The number of columns of the layout.

overlay ['dims' or 'data_vars', default 'dims'] If 'dims', make one figure for each data variable and overlay the dimensions. If 'data_vars', make one figure for each dimension and overlay the data variables. In the latter case, all variables must have the same dimensions.

tooltips [dict, optional] Names of tooltips mapping to glyph properties or source columns, e.g. `{'datetime': '$x{%F %T.%3N}'}`.

tools [str, optional] bokeh tool string.

palette [iterable, optional] The palette to use when overlaying multiple glyphs.

ignore_index [bool, default False] If True, replace the x-axis values of the data by an appropriate evenly spaced index.

Methods

__init__(data, x[, overlay, coords, glyphs, ...])	Constructor.
add_annotation(annotation[, onto])	Add an annotation to a figure in the layout.
add_figure(data[, glyphs, coords, name])	Add a figure to the layout.
add_overlay(data[, glyphs, coords, name, onto])	Add an overlay to a figure in the layout.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
modify_figures(modifiers[, figures])	Modify the attributes of a figure.
show([filename, remake_layout])	Show the plot in an HTML file.

Attributes

default_tools

3.2.2 xrview.html.HtmlGridPlot

class xrview.html.**HtmlGridPlot**(*panels*, *ncols=1*, *toolbar_location='above'*)

An HTML grid plot.

__init__(*panels*, *ncols=1*, *toolbar_location='above'*)

Constructor.

Methods

__init__(panels[, ncols, toolbar_location])	Constructor.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
show([filename, remake_layout])	Show the plot in an HTML file.

3.2.3 xrview.html.HtmlSpacer

class xrview.html.**HtmlSpacer**

An HTML spacer.

__init__()

Constructor.

Methods

__init__()	Constructor.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
show(*args, **kwargs)	Show the layout.

3.3 Notebook

Module: xrview.notebook

NotebookPlot	Base class for notebook plots.
NotebookGridPlot	A notebook grid plot.
NotebookSpacer	A notebook spacer.
NotebookViewer	Base class for notebook viewers.
NotebookTimeseriesViewer	A notebook time-series viewer.

3.3.1 xrview.notebook.NotebookPlot

class xrview.notebook.**NotebookPlot**(*data*, *x*, *overlay='dims'*, *coords=None*, *glyphs='line'*, *title=None*, *share_y=False*, *tooltips=None*, *tools=None*, *toolbar_location='right'*, *figsize=600, 300*, *ncols=1*, *palette=None*, *ignore_index=False*, *theme=None*, ***fig_kwargs*)

Base class for notebook plots.

__init__(*data*, *x*, *overlay='dims'*, *coords=None*, *glyphs='line'*, *title=None*, *share_y=False*, *tooltips=None*, *tools=None*, *toolbar_location='right'*, *figsize=600, 300*, *ncols=1*, *palette=None*, *ignore_index=False*, *theme=None*, ***fig_kwargs*)

Constructor.

Parameters

data [xarray DataArray or Dataset] The data to display.

x [str] The name of the dimension in data that contains the x-axis values.

glyphs [str, BaseGlyph or iterable, default 'line'] The glyph to use for plotting.

figsize [iterable, default (600, 300)] The size of the figure in pixels.

ncols [int, default 1] The number of columns of the layout.

overlay ['dims' or 'data_vars', default 'dims'] If 'dims', make one figure for each data variable and overlay the dimensions. If 'data_vars', make one figure for each dimension and overlay the data variables. In the latter case, all variables must have the same dimensions.

tooltips [dict, optional] Names of tooltips mapping to glyph properties or source columns, e.g. {'datetime': '$x{%F %T.%3N}'}.

tools [str, optional] bokeh tool string.

palette [iterable, optional] The palette to use when overlaying multiple glyphs.

ignore_index [bool, default False] If True, replace the x-axis values of the data by an appropriate evenly spaced index.

Methods

__init__(data, x[, overlay, coords, glyphs, ...])	Constructor.
add_annotation(annotation[, onto])	Add an annotation to a figure in the layout.
add_figure(data[, glyphs, coords, name])	Add a figure to the layout.
add_overlay(data[, glyphs, coords, name, onto])	Add an overlay to a figure in the layout.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
modify_figures(modifiers[, figures])	Modify the attributes of a figure.
show([remake_layout])	Show the plot in a jupyter notebook.

Attributes

default_tools

3.3.2 xrview.notebook.NotebookGridPlot

class xrview.notebook.**NotebookGridPlot**(*panels*, *ncols=1*, *toolbar_location='above'*)

A notebook grid plot.

__init__(*panels*, *ncols=1*, *toolbar_location='above'*)

Constructor.

Methods

__init__(panels[, ncols, toolbar_location])	Constructor.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
show([remake_layout])	Show the plot in a jupyter notebook.

3.3.3 xrview.notebook.NotebookSpacer

class xrview.notebook.**NotebookSpacer**

A notebook spacer.

__init__()

Constructor.

Methods

__init__()	Constructor.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
show(*args, **kwargs)	Show the layout.

3.3.4 xrview.notebook.NotebookViewer

class xrview.notebook.**NotebookViewer**(**args*, ***kwargs*)

Base class for notebook viewers.

__init__(**args*, ***kwargs*)

Constructor.

Parameters

data [xarray DataArray or Dataset] The data to display.

x [str] The name of the dimension in `data` that contains the x-axis values.

glyphs [str, BaseGlyph or iterable, default 'line'] The glyph to use for plotting.

figsize [iterable, default (600, 300)] The size of the figure in pixels.

ncols [int, default 1] The number of columns of the layout.

overlay ['dims' or 'data_vars', default 'dims'] If 'dims', make one figure for each data variable and overlay the dimensions. If 'data_vars', make one figure for each dimension and overlay the data variables. In the latter case, all variables must have the same dimensions.

tooltips [dict, optional] Names of tooltips mapping to glyph properties or source columns, e.g. `{'datetime': '$x{%F %T.%3N}'}`.

tools [str, optional] bokeh tool string.

palette [iterable, optional] The palette to use when overlaying multiple glyphs.

ignore_index [bool, default False] If True, replace the x-axis values of the data by an appropriate evenly spaced index.

Methods

__init__(*args, **kwargs)	Constructor.
add_annotation(annotation[, onto])	Add an annotation to a figure in the layout.
add_figure(data[, glyphs, coords, name])	Add a figure to the layout.
add_interaction(interaction)	Add an interaction to the layout.
add_overlay(data[, glyphs, coords, name, onto])	Add an overlay to a figure in the layout.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
modify_figures(modifiers[, figures])	Modify the attributes of a figure.

continues on next page

Table 12 – continued from previous page

on_reset(event)	Callback for reset event.
on_selected_points_change(attr, old, new)	Callback for selection event.
reset_handlers()	Reset handlers.
show([notebook_url, port, remake_layout, ...])	Show the app in a jupyter notebook.
update_handler(handler)	Update a single handler.
update_handlers([handlers])	Update handlers.
update_inplace(other)	Update this instance with the properties of another layout.

Attributes

default_tools

3.3.5 xrview.notebook.NotebookTimeseriesViewer

class xrview.notebook.**NotebookTimeseriesViewer**(*data*, *x*, *overlay='dims'*, *glyphs='line'*, *tooltips=None*, *tools=None*, *figsize=600, 300*, *ncols=1*, *palette=None*, *ignore_index=False*, *resolution=4*, *max_workers=10*, *lowpass=False*, *verbose=0*, ***fig_kwargs*)

A notebook time-series viewer.

__init__(*data*, *x*, *overlay='dims'*, *glyphs='line'*, *tooltips=None*, *tools=None*, *figsize=600, 300*, *ncols=1*, *palette=None*, *ignore_index=False*, *resolution=4*, *max_workers=10*, *lowpass=False*, *verbose=0*, ***fig_kwargs*)

Constructor.

Parameters

data [xarray DataArray or Dataset] The data to display.

x [str] The name of the dimension in `data` that contains the x-axis values.

glyphs [str, BaseGlyph or iterable, default 'line'] The glyph to use for plotting.

figsize [iterable, default (600, 300)] The size of the figure in pixels.

ncols [int, default 1] The number of columns of the layout.

overlay ['dims' or 'data_vars', default 'dims'] If 'dims', make one figure for each data variable and overlay the dimensions. If 'data_vars', make one figure for each dimension and overlay the data variables. In the latter case, all variables must have the same dimensions.

tooltips [dict, optional] Names of tooltips mapping to glyph properties or source columns, e.g. {'datetime': '@index{%F %T.%3N}'}.

tools [str, optional] bokeh tool string.

palette [iterable, optional] The palette to use when overlaying multiple glyphs.

ignore_index [bool, default Falseh] If True, replace the x-axis values of the data by an appropriate evenly spaced index.

resolution [int, default 4] The number of points to render for each pixel.

max_workers [int, default 10] The maximum number of workers in the thread pool to perform the down-sampling.

lowpass [bool, default False] If True, filter the values with a low-pass filter before down-sampling.

verbose [int, default 0] The level of verbosity.

Methods

__init__(data, x[, overlay, glyphs, ...])	Constructor.
add_annotation(annotation[, onto])	Add an annotation to a figure in the layout.
add_figure(data[, glyphs, coords, name, ...])	Add a figure to the layout.
add_interaction(interaction)	Add an interaction to the layout.
add_overlay(data[, glyphs, coords, name, ...])	Add an overlay to a figure in the layout.
copy([with_data])	Create a copy of this instance.
export(filename[, mode])	Export the layout as as png or svg file.
make_doc()	Make the document.
make_layout()	Make the layout.
modify_figures(modifiers[, figures])	Modify the attributes of a figure.
on_reset(event)	Callback for reset event.
on_selected_points_change(attr, old, new)	Callback for selection event.
on_xrange_change(attr, old, new)	Callback for xrange change event.
reset_handlers()	Reset handlers.
show([notebook_url, port, remake_layout, ...])	Show the app in a jupyter notebook.
update_handler(handler)	Update a single handler.
update_handlers([handlers])	Update handlers.
update_inplace(other)	Update this instance with the properties of another layout.

Attributes

default_tools

3.4 Glyphs

Module: xrview.glyphs

Line	A line glyph.
Circle	A circle glyph.
Diamond	A diamond glyph.
Square	A square glyph.
Triangle	A triangle glyph.
Ray	A ray glyph.
HBar	A horizontal bar glyph.
VBar	A vertical bar glyph.
Rect	A rectangle glyph.
Whisker	A whisker annotation.

continues on next page

Table 16 – continued from previous page

Band	A band annotation.
VLine	A collection of vertical lines.
ErrorLine	A line with an error bar.
ErrorCircle	A circle with an error bar.
BoxWhisker	A box-whisker glyph.

3.4.1 xrview.glyphs.Line

class xrview.glyphs.**Line**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A line glyph.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.2 xrview.glyphs.Circle

class xrview.glyphs.**Circle**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A circle glyph.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

`__init__`([x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.3 xrview.glyphs.Diamond

class xrview.glyphs.**Diamond**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A diamond glyph.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

`__init__`([x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.4 xrview.glyphs.Square

class xrview.glyphs.**Square**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A square glyph.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.5 xrview.glyphs.Triangle

class xrview.glyphs.**Triangle**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A triangle glyph.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.6 xrview.glyphs.Ray

class xrview.glyphs.**Ray**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A ray glyph.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.7 xrview.glyphs.HBar

class xrview.glyphs.**HBar**(*height*, *x_arg='right'*, *y_arg='y'*, *other=0.0*, ***kwargs*)

A horizontal bar glyph.

__init__(*height*, *x_arg='right'*, *y_arg='y'*, *other=0.0*, ***kwargs*)

Constructor.

Parameters

height [str or float] The name of a coordinate or a fixed value that will represent the height of the bar.

x_arg: str, default 'right' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

other: str or float, default 0. The name of a coordinate or a fixed value that will represent the other side of the bar (i.e. the left side when x_arg='right').

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__(height[, x_arg, y_arg, other])	Constructor.

Attributes

default_kwargs
method

3.4.8 xrview.glyphs.VBar

class xrview.glyphs.**VBar**(*width*, *x_arg='x'*, *y_arg='top'*, *other=0.0*, ***kwargs*)
A vertical bar glyph.

__init__(*width*, *x_arg='x'*, *y_arg='top'*, *other=0.0*, ***kwargs*)
Constructor.

Parameters

width [str or float] The name of a coordinate or a fixed value that will represent the width of the bar.

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'top' The glyph argument associated with y-axis values in the data.

other: str or float, default 0. The name of a coordinate or a fixed value that will represent the other side of the bar (i.e. the bottom side when y_arg='top').

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__(width[, x_arg, y_arg, other])	Constructor.

Attributes

default_kwargs
method

3.4.9 xrview.glyphs.Rect

class xrview.glyphs.**Rect**(*width*, *height*, *x_arg='x'*, *y_arg='y'*, ***kwargs*)
A rectangle glyph.

__init__(*width*, *height*, *x_arg='x'*, *y_arg='y'*, ***kwargs*)
Constructor.

Parameters

width [str or float] The name of a coordinate or a fixed value that will represent the width of the rectangle.

height [str or float] The name of a coordinate or a fixed value that will represent the height of the rectangle.

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__(width, height[, x_arg, y_arg])	Constructor.

Attributes

default_kwargs
method

3.4.10 xrview.glyphs.Whisker

class xrview.glyphs.**Whisker**(*x_arg='base'*, *y_arg='upper'*, *other=0.0*, ***kwargs*)

A whisker annotation.

__init__(*x_arg='base'*, *y_arg='upper'*, *other=0.0*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'base' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'upper' The glyph argument associated with y-axis values in the data.

other: str or float, default 0. The name of a coordinate or a fixed value that will represent the other end of the whisker (i.e. the lower end when y_arg='upper').

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg, other])	Constructor.

Attributes

default_kwargs

3.4.11 xrview.glyphs.Band

class xrview.glyphs.**Band**(*x_arg='base'*, *y_arg='upper'*, *other=0.0*, ***kwargs*)

A band annotation.

__init__(*x_arg='base'*, *y_arg='upper'*, *other=0.0*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'base' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'upper' The glyph argument associated with y-axis values in the data.

other: str or float, default 0. The name of a coordinate or a fixed value that will represent the other end of the whisker (i.e. the lower end when y_arg='upper').

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg, other])	Constructor.

Attributes

default_kwargs

3.4.12 xrview.glyphs.VLine

class xrview.glyphs.**VLine**(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

A collection of vertical lines.

__init__(*x_arg='x'*, *y_arg='y'*, ***kwargs*)

Constructor.

Parameters

x_arg: str, default 'x' The glyph argument associated with x-axis values in the data.

y_arg: str, default 'y' The glyph argument associated with y-axis values in the data.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__([x_arg, y_arg])	Constructor.

Attributes

default_kwargs

3.4.13 xrview.glyphs.ErrorLine

class xrview.glyphs.**ErrorLine**(*lower*, *upper*, ***kwargs*)

A line with an error bar.

__init__(*lower*, *upper*, ***kwargs*)

Constructor.

Parameters

lower: str or float The name of a coordinate or a fixed value that will represent the lower end of the error bar.

upper: str or float The name of a coordinate or a fixed value that will represent the upper end of the error bar.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

`__init__`(lower, upper, **kwargs)	Constructor.

Attributes

`default_kwargs`

3.4.14 xrview.glyphs.ErrorCircle

class xrview.glyphs.**ErrorCircle**(*lower*, *upper*, ***kwargs*)

A circle with an error bar.

__init__(*lower*, *upper*, ***kwargs*)

Constructor.

Parameters

lower: str or float The name of a coordinate or a fixed value that will represent the lower end of the error bar.

upper: str or float The name of a coordinate or a fixed value that will represent the upper end of the error bar.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

`__init__`(lower, upper, **kwargs)	Constructor.

Attributes

`default_kwargs`

3.4.15 xrview.glyphs.BoxWhisker

class xrview.glyphs.**BoxWhisker**(*width*, *q_lower*, *w_lower*, *q_upper*, *w_upper*, ***kwargs*)
A box-whisker glyph.

__init__(*width*, *q_lower*, *w_lower*, *q_upper*, *w_upper*, ***kwargs*)
Constructor.

Parameters

width [str or float] The name of a coordinate or a fixed value that will represent the width of the box.

q_lower: str or float The name of a coordinate or a fixed value that will represent the lower end of the box.

w_lower: str or float The name of a coordinate or a fixed value that will represent the lower end of the error bar.

q_upper: str or float The name of a coordinate or a fixed value that will represent the upper end of the box.

w_upper: str or float The name of a coordinate or a fixed value that will represent the upper end of the error bar.

kwargs: Additional keyword arguments to be passed to the underlying bokeh glyph(s).

Methods

__init__(width, q_lower, w_lower, q_upper, ...)	Constructor.

Attributes

default_kwargs

3.5 Interactions

Module: xrview.interactions

CoordValSelect	A list widget for selecting unique values of a certain coordinate.

3.5.1 xrview.interactions.CoordValSelect

class xrview.interactions.**CoordValSelect**(*coord*, *max_elements=30*, *location='right'*)
A list widget for selecting unique values of a certain coordinate.

__init__(*coord*, *max_elements=30*, *location='right'*)
Constructor.

Parameters

coord: str The name of the coordinate.

max_elements: int, default 30 The maximum size of the displayed list of unique values.

location: str, default 'right' The location of the widget in the layout.

Methods

__init__(coord[, max_elements, location])	Constructor.
attach(context)	Attach element to context.
collect_hook(data)	Hook for context.collect.
layout_hook()	Hook for layout creation.
on_selected_coord_change(attr, old, new)	Callback for multi-select change event.

3.6 Class member details

3.6.1 html

HtmlPlot

Plot modifiers

HtmlPlot.add_annotation	Add an annotation to a figure in the layout.
HtmlPlot.add_figure	Add a figure to the layout.
HtmlPlot.add_overlay	Add an overlay to a figure in the layout.
HtmlPlot.modify_figures	Modify the attributes of a figure.

xrview.html.HtmlPlot.add_annotation

HtmlPlot.**add_annotation**(*annotation*, *onto=None*)
Add an annotation to a figure in the layout.

Parameters

annotation :

onto [str or int, optional] Title or index of the figure on which the annotation will be overlaid. By default, the annotation is overlaid on all figures.

xrview.html.HtmlPlot.add_figure

HtmlPlot.**add_figure**(*data*, *glyphs='line'*, *coords=None*, *name=None*)

Add a figure to the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used as the title of the figure. If not provided, the name of the DataArray will be used.

xrview.html.HtmlPlot.add_overlay

HtmlPlot.**add_overlay**(*data*, *glyphs='line'*, *coords=None*, *name=None*, *onto=None*)

Add an overlay to a figure in the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used to identify the overlay. If not provided, the name of the DataArray will be used.

onto [str or int, optional] Title or index of the figure on which the element will be overlaid. By default, the element is overlaid on all figures.

xrview.html.HtmlPlot.modify_figures

HtmlPlot.**modify_figures**(*modifiers*, *figures=None*)

Modify the attributes of a figure.

Parameters

modifiers [dict] The attributes to modify. Keys can reference sub-attributes, e.g. 'xaxis.axis_label'.

figures [int or iterable of int, optional] The index(es) of the figure(s) to modify.

Show and export

HtmlPlot.export	Export the layout as as png or svg file.
HtmlPlot.show	Show the plot in an HTML file.

xrview.html.HtmlPlot.export

HtmlPlot.**export**(*filename*, *mode='auto'*)

Export the layout as as png or svg file.

Parameters

filename [str] The path of the exported file.

mode ['auto', 'png' or 'svg', default 'auto'] Whether to export as png or svg. Note that multi-figure layouts will be split into individual files for each figure in the svg mode. 'auto' will try to determine the mode automatically from the file extension.

xrview.html.HtmlPlot.show

HtmlPlot.**show**(*filename=None*, *remake_layout=False*)

Show the plot in an HTML file.

Parameters

filename [str, optional] If specified, save the plot to this HTML file.

remake_layout [bool, default False] If True, call `make_layout` even when the layout has already been created. Note that any changes made by `modify_figures` will be omitted.

Internal

HtmlPlot.copy	Create a copy of this instance.
HtmlPlot.make_doc	Make the document.
HtmlPlot.make_layout	Make the layout.

xrview.html.HtmlPlot.copy

HtmlPlot.**copy**(*with_data=False*)

Create a copy of this instance.

Parameters

with_data [bool, default False] If true, also copy the data.

Returns

new [xrview.core.panel.BasePanel] The copied object.

xrview.html.HtmlPlot.make_doc

HtmlPlot.**make_doc**()
Make the document.

xrview.html.HtmlPlot.make_layout

HtmlPlot.**make_layout**()
Make the layout.

3.6.2 notebook

NotebookPlot

Plot modifiers

NotebookPlot.add_annotation	Add an annotation to a figure in the layout.
NotebookPlot.add_figure	Add a figure to the layout.
NotebookPlot.add_overlay	Add an overlay to a figure in the layout.
NotebookPlot.modify_figures	Modify the attributes of a figure.

xrview.notebook.NotebookPlot.add_annotation

NotebookPlot.**add_annotation**(*annotation*, *onto=None*)
Add an annotation to a figure in the layout.

Parameters

annotation :

onto [str or int, optional] Title or index of the figure on which the annotation will be overlaid. By default, the annotation is overlaid on all figures.

xrview.notebook.NotebookPlot.add_figure

NotebookPlot.**add_figure**(*data*, *glyphs='line'*, *coords=None*, *name=None*)
Add a figure to the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used as the title of the figure. If not provided, the name of the DataArray will be used.

xrview.notebook.NotebookPlot.add_overlay

NotebookPlot.**add_overlay**(*data*, *glyphs='line'*, *coords=None*, *name=None*, *onto=None*)

Add an overlay to a figure in the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used to identify the overlay. If not provided, the name of the DataArray will be used.

onto [str or int, optional] Title or index of the figure on which the element will be overlaid. By default, the element is overlaid on all figures.

xrview.notebook.NotebookPlot.modify_figures

NotebookPlot.**modify_figures**(*modifiers*, *figures=None*)

Modify the attributes of a figure.

Parameters

modifiers [dict] The attributes to modify. Keys can reference sub-attributes, e.g. 'xaxis.axis_label'.

figures [int or iterable of int, optional] The index(es) of the figure(s) to modify.

Show and export

NotebookPlot.export	Export the layout as as png or svg file.
NotebookPlot.show	Show the plot in a jupyter notebook.

xrview.notebook.NotebookPlot.export

NotebookPlot.**export**(*filename*, *mode='auto'*)

Export the layout as as png or svg file.

Parameters

filename [str] The path of the exported file.

mode ['auto', 'png' or 'svg', default 'auto'] Whether to export as png or svg. Note that multi-figure layouts will be split into individual files for each figure in the svg mode. 'auto' will try to determine the mode automatically from the file extension.

xrview.notebook.NotebookPlot.show

NotebookPlot.**show**(*remake_layout=False*)
Show the plot in a jupyter notebook.

Parameters

remake_layout [bool, default False] If True, call `make_layout` even when the layout has already been created. Note that any changes made by `modify_figures` will be omitted.

Internal

NotebookPlot.copy	Create a copy of this instance.
NotebookPlot.make_doc	Make the document.
NotebookPlot.make_layout	Make the layout.

xrview.notebook.NotebookPlot.copy

NotebookPlot.**copy**(*with_data=False*)
Create a copy of this instance.

Parameters

with_data [bool, default False] If true, also copy the data.

Returns

new [xrview.core.panel.BasePanel] The copied object.

xrview.notebook.NotebookPlot.make_doc

NotebookPlot.**make_doc**()
Make the document.

xrview.notebook.NotebookPlot.make_layout

NotebookPlot.**make_layout**()
Make the layout.

NotebookViewer

Plot modifiers

NotebookViewer.add_annotation	Add an annotation to a figure in the layout.
NotebookViewer.add_figure	Add a figure to the layout.
NotebookViewer.add_interaction	Add an interaction to the layout.
NotebookViewer.add_overlay	Add an overlay to a figure in the layout.
NotebookViewer.modify_figures	Modify the attributes of a figure.

xrview.notebook.NotebookViewer.add_annotation

NotebookViewer.**add_annotation**(*annotation*, *onto=None*)

Add an annotation to a figure in the layout.

Parameters

annotation :

onto [str or int, optional] Title or index of the figure on which the annotation will be overlaid. By default, the annotation is overlaid on all figures.

xrview.notebook.NotebookViewer.add_figure

NotebookViewer.**add_figure**(*data*, *glyphs='line'*, *coords=None*, *name=None*)

Add a figure to the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used as the title of the figure. If not provided, the name of the DataArray will be used.

xrview.notebook.NotebookViewer.add_interaction

NotebookViewer.**add_interaction**(*interaction*)

Add an interaction to the layout.

Parameters

interaction [xrview.interactions.BaseInteraction] The interaction to add.

xrview.notebook.NotebookViewer.add_overlay

NotebookViewer.**add_overlay**(*data*, *glyphs='line'*, *coords=None*, *name=None*, *onto=None*)

Add an overlay to a figure in the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used to identify the overlay. If not provided, the name of the DataArray will be used.

onto [str or int, optional] Title or index of the figure on which the element will be overlaid. By default, the element is overlaid on all figures.

xrview.notebook.NotebookViewer.modify_figures

NotebookViewer.**modify_figures**(*modifiers*, *figures=None*)

Modify the attributes of a figure.

Parameters

modifiers [dict] The attributes to modify. Keys can reference sub-attributes, e.g. 'xaxis.axis_label'.

figures [int or iterable of int, optional] The index(es) of the figure(s) to modify.

Show and export

NotebookViewer.export	Export the layout as as png or svg file.
NotebookViewer.show	Show the app in a jupyter notebook.

xrview.notebook.NotebookViewer.export

NotebookViewer.**export**(*filename*, *mode='auto'*)

Export the layout as as png or svg file.

Parameters

filename [str] The path of the exported file.

mode ['auto', 'png' or 'svg', default 'auto'] Whether to export as png or svg. Note that multi-figure layouts will be split into individual files for each figure in the svg mode. 'auto' will try to determine the mode automatically from the file extension.

xrview.notebook.NotebookViewer.show

NotebookViewer.**show**(*notebook_url=None*, *port=0*, *remake_layout=False*, *verbose=False*)

Show the app in a jupyter notebook.

Parameters

notebook_url [str, optional] The URL of the notebook. Will be determined automatically if not specified.

port [int, default 0] The port over which the app will be served. Chosen randomly if set to 0.

remake_layout [bool, default False] If True, call `make_layout` even when the layout has already been created. Note that any changes made by `modify_figures` will be omitted.

verbose [bool, default False] If True, create the document once again outside of show_app in order to show errors.

Internal

NotebookViewer.copy	Create a copy of this instance.
NotebookViewer.make_doc	Make the document.
NotebookViewer.make_layout	Make the layout.
NotebookViewer.reset_handlers	Reset handlers.
NotebookViewer.update_handler	Update a single handler.
NotebookViewer.update_handlers	Update handlers.
NotebookViewer.update_inplace	Update this instance with the properties of another layout.

xrview.notebook.NotebookViewer.copy

NotebookViewer.**copy**(*with_data=False*)
Create a copy of this instance.

Parameters

with_data [bool, default False] If true, also copy the data.

Returns

new [xrview.core.panel.BasePanel] The copied object.

xrview.notebook.NotebookViewer.make_doc

NotebookViewer.**make_doc**()
Make the document.

xrview.notebook.NotebookViewer.make_layout

NotebookViewer.**make_layout**()
Make the layout.

xrview.notebook.NotebookViewer.reset_handlers

NotebookViewer.**reset_handlers**()
Reset handlers.

xrview.notebook.NotebookViewer.update_handler

NotebookViewer.**update_handler**(*handler*)
Update a single handler.

xrview.notebook.NotebookViewer.update_handlers

NotebookViewer.**update_handlers**(*handlers=None*)
Update handlers.

xrview.notebook.NotebookViewer.update_inplace

NotebookViewer.**update_inplace**(*other*)
Update this instance with the properties of another layout.

Parameters

other [xrview.core.viewer.BaseViewer] The instance that replaces the current instance.

Callbacks

NotebookViewer.on_reset	Callback for reset event.
NotebookViewer.on_selected_points_change	Callback for selection event.

xrview.notebook.NotebookViewer.on_reset

NotebookViewer.**on_reset**(*event*)
Callback for reset event.

xrview.notebook.NotebookViewer.on_selected_points_change

NotebookViewer.**on_selected_points_change**(*attr*, *old*, *new*)
Callback for selection event.

NotebookTimeseriesViewer

Plot modifiers

NotebookTimeseriesViewer.add_annotation	Add an annotation to a figure in the layout.
NotebookTimeseriesViewer.add_figure	Add a figure to the layout.
NotebookTimeseriesViewer.add_interaction	Add an interaction to the layout.
NotebookTimeseriesViewer.add_overlay	Add an overlay to a figure in the layout.
NotebookTimeseriesViewer.modify_figures	Modify the attributes of a figure.

xrview.notebook.NotebookTimeseriesViewer.add_annotation

NotebookTimeseriesViewer.**add_annotation**(*annotation*, *onto=None*)

Add an annotation to a figure in the layout.

Parameters

annotation :

onto [str or int, optional] Title or index of the figure on which the annotation will be overlaid. By default, the annotation is overlaid on all figures.

xrview.notebook.NotebookTimeseriesViewer.add_figure

NotebookTimeseriesViewer.**add_figure**(*data*, *glyphs='line'*, *coords=None*, *name=None*, *resolution=None*)

Add a figure to the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used as the title of the figure. If not provided, the name of the DataArray will be used.

resolution [float, optional] The number of points to render for each pixel of this figure. If not specified, the resolution of the viewer is used.

xrview.notebook.NotebookTimeseriesViewer.add_interaction

NotebookTimeseriesViewer.**add_interaction**(*interaction*)

Add an interaction to the layout.

Parameters

interaction [xrview.interactions.BaseInteraction] The interaction to add.

xrview.notebook.NotebookTimeseriesViewer.add_overlay

NotebookTimeseriesViewer.**add_overlay**(*data*, *glyphs='line'*, *coords=None*, *name=None*, *onto=None*, *resolution=None*)

Add an overlay to a figure in the layout.

Parameters

data [xarray.DataArray] The data to display.

glyphs [str, BaseGlyph or iterable thereof, default 'line'] The glyph (or glyphs) to display.

coords [iterable of str, optional] The coordinates of the DataArray to include. This is necessary for composite glyphs such as BoxWhisker.

name [str, optional] The name of the DataArray which will be used as the title of the figure. If not provided, the name of the DataArray will be used.

onto [str or int, optional] Title or index of the figure on which the element will be overlaid. By default, the element is overlaid on all figures.

resolution [float, optional] The number of points to render for each pixel of this figure. If not specified, the resolution of the viewer is used.

xrview.notebook.NotebookTimeseriesViewer.modify_figures

`NotebookTimeseriesViewer.`**`modify_figures`**(*modifiers*, *figures=None*)

Modify the attributes of a figure.

Parameters

modifiers [dict] The attributes to modify. Keys can reference sub-attributes, e.g. 'xaxis.axis_label'.

figures [int or iterable of int, optional] The index(es) of the figure(s) to modify.

Show and export

NotebookTimeseriesViewer.export	Export the layout as as png or svg file.
NotebookTimeseriesViewer.show	Show the app in a jupyter notebook.

xrview.notebook.NotebookTimeseriesViewer.export

`NotebookTimeseriesViewer.`**`export`**(*filename*, *mode='auto'*)

Export the layout as as png or svg file.

Parameters

filename [str] The path of the exported file.

mode ['auto', 'png' or 'svg', default 'auto'] Whether to export as png or svg. Note that multi-figure layouts will be split into individual files for each figure in the svg mode. 'auto' will try to determine the mode automatically from the file extension.

xrview.notebook.NotebookTimeseriesViewer.show

`NotebookTimeseriesViewer.`**`show`**(*notebook_url=None*, *port=0*, *remake_layout=False*, *verbose=False*)

Show the app in a jupyter notebook.

Parameters

notebook_url [str, optional] The URL of the notebook. Will be determined automatically if not specified.

port [int, default 0] The port over which the app will be served. Chosen randomly if set to 0.

remake_layout [bool, default False] If True, call `make_layout` even when the layout has already been created. Note that any changes made by `modify_figures` will be omitted.

verbose [bool, default False] If True, create the document once again outside of show_app in order to show errors.

Internal

NotebookTimeseriesViewer.copy	Create a copy of this instance.
NotebookTimeseriesViewer.make_doc	Make the document.
NotebookTimeseriesViewer.make_layout	Make the layout.
NotebookTimeseriesViewer.reset_handlers	Reset handlers.
NotebookTimeseriesViewer.update_handler	Update a single handler.
NotebookTimeseriesViewer.update_handlers	Update handlers.
NotebookTimeseriesViewer.update_inplace	Update this instance with the properties of another layout.

xrview.notebook.NotebookTimeseriesViewer.copy

NotebookTimeseriesViewer.**copy**(*with_data=False*)

Create a copy of this instance.

Parameters

with_data [bool, default False] If true, also copy the data.

Returns

new [xrview.core.panel.BasePanel] The copied object.

xrview.notebook.NotebookTimeseriesViewer.make_doc

NotebookTimeseriesViewer.**make_doc**()

Make the document.

xrview.notebook.NotebookTimeseriesViewer.make_layout

NotebookTimeseriesViewer.**make_layout**()

Make the layout.

xrview.notebook.NotebookTimeseriesViewer.reset_handlers

NotebookTimeseriesViewer.**reset_handlers**()

Reset handlers.

xrview.notebook.NotebookTimeseriesViewer.update_handler

NotebookTimeseriesViewer.**update_handler**(*handler*)
Update a single handler.

xrview.notebook.NotebookTimeseriesViewer.update_handlers

NotebookTimeseriesViewer.**update_handlers**(*handlers=None*)
Update handlers.

xrview.notebook.NotebookTimeseriesViewer.update_inplace

NotebookTimeseriesViewer.**update_inplace**(*other*)
Update this instance with the properties of another layout.

Parameters

other [xrview.core.viewer.BaseViewer] The instance that replaces the current instance.

Callbacks

NotebookTimeseriesViewer.on_reset	Callback for reset event.
NotebookTimeseriesViewer.on_selected_points_change	Callback for selection event.
NotebookTimeseriesViewer.on_xrange_change	Callback for xrange change event.

xrview.notebook.NotebookTimeseriesViewer.on_reset

NotebookTimeseriesViewer.**on_reset**(*event*)
Callback for reset event.

xrview.notebook.NotebookTimeseriesViewer.on_selected_points_change

NotebookTimeseriesViewer.**on_selected_points_change**(*attr*, *old*, *new*)
Callback for selection event.

xrview.notebook.NotebookTimeseriesViewer.on_xrange_change

NotebookTimeseriesViewer.**on_xrange_change**(*attr*, *old*, *new*)
Callback for xrange change event.

CHAPTER

FOUR

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/phausamann/xrview/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/phausamann/xrview/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/xrview.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 xrview
$ cd xrview/
$ 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 all tests and linters with tox:

```
$ tox
```

   To get tox, just pip install it 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.

4.3 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 3.6, 3,7 and 3.8. Check https://travis-ci.org/phausamann/xrview/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_xrview
```

4.5 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:

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

Travis will then deploy to PyPI if tests pass.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Peter Hausamann <peter.hausamann@tum.de>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 0.2.1 (Oct 19, 2020)

6.1.1 Bug fixes

- Fixed tooltip rendering for datetime axis.

6.2 0.2.0 (Sep 8, 2020)

6.2.1 Breaking changes

- Dropped compatibility with Python 2.7 and 3.5 and bokeh versions < 2.2

6.3 0.1.0 (Sep 8, 2020)

- First official release
- Last release compatible with Python 2.7/3.5

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

A

B

C

D

E