

---

**XrViz**

***Release 0.1.0+119.g7b933f2.dirty***

**Harman Deep Singh, Rich Signell, Martin Durant**

**Sep 04, 2019**



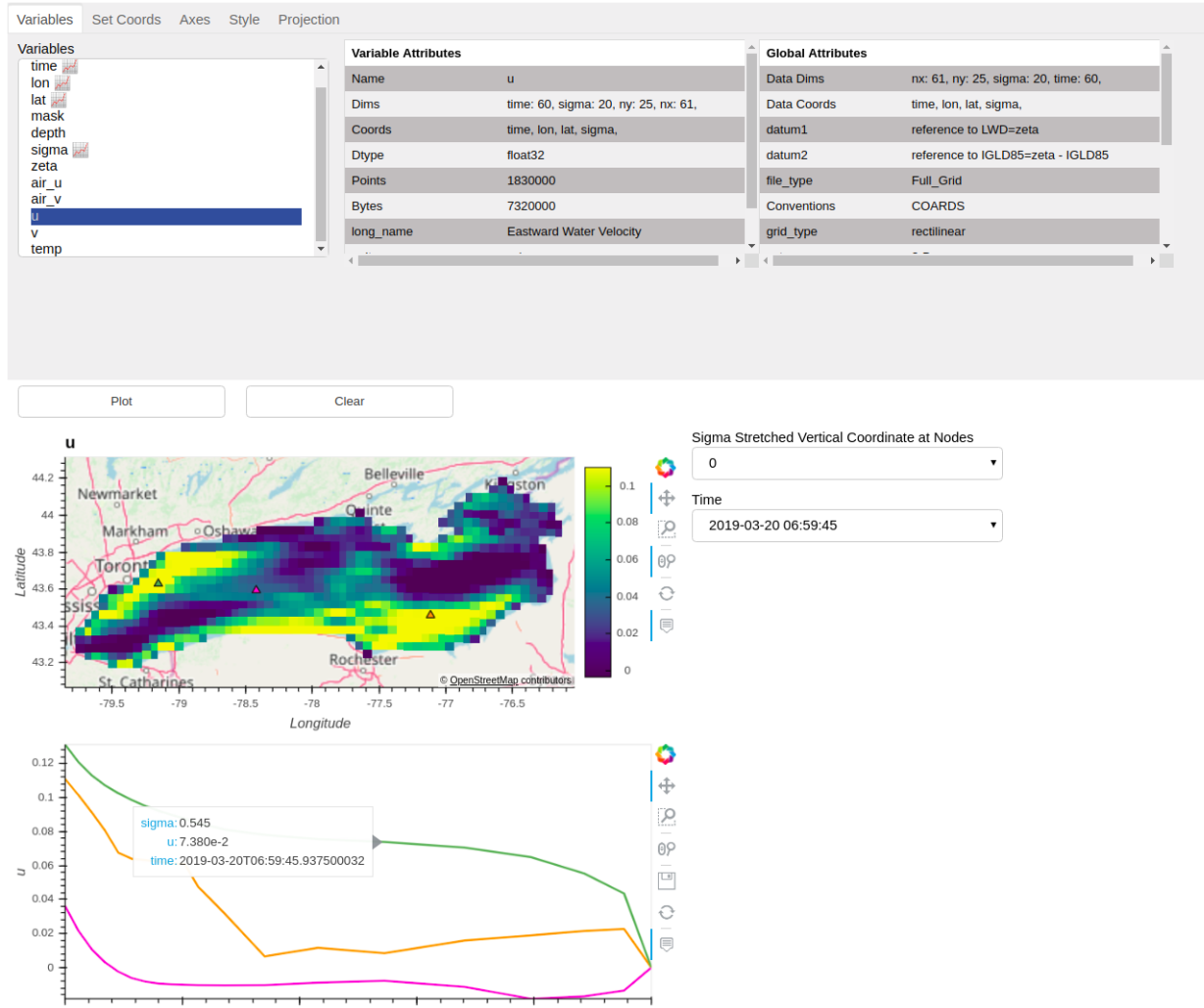
# DOCUMENTATION CONTENTS

<b>1</b>	<b>Quickstart</b>	<b>3</b>
1.1	Installation . . . . .	3
1.2	Example . . . . .	3
1.3	Overview . . . . .	4
<b>2</b>	<b>Graphical User Interface</b>	<b>5</b>
2.1	Variables . . . . .	5
2.2	Set_Coords . . . . .	5
2.3	Axes . . . . .	6
2.4	Style . . . . .	8
2.5	Projection . . . . .	8
<b>3</b>	<b>Command Line Interface</b>	<b>9</b>
<b>4</b>	<b>Set Initial Parameters</b>	<b>11</b>
<b>5</b>	<b>API</b>	<b>13</b>
5.1	Variables . . . . .	13
5.2	Set_Coords . . . . .	14
5.3	Axes . . . . .	14
5.4	Style . . . . .	15
5.5	Projection . . . . .	16
5.6	Control . . . . .	16
5.7	Dashboard . . . . .	17
5.8	Example . . . . .	19
<b>6</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



### An interactive visualisation interface for Xarrays

XrViz is an interactive graphical user interface(GUI) for visually browsing Xarrays. You can view data arrays along various dimensions, examine data values, change color maps, extract series, display geographic data on maps and much more. It is built on [Xarray](#), [HvPlot](#) and [Panel](#). It can be used with [Intake](#) to ease the process of investigating and loading datasets.



It offers the following functionality:

1. Quick visualization of xarray data
2. Customization programatically or interactively via widgets
3. Mapping of geospatial datasets using [Geoviews](#).
4. Automatic determination of geospatial coordinates for CF-Convention data using [Metpy](#)
5. Optional aggregation along non-map dimensions (e.g. mean over the time dimension)
6. Interactive extraction of series plots for non-mapped dimensions (e.g. time series at a point)



## QUICKSTART

### 1.1 Installation

If you are using [Anaconda](#) or [Miniconda](#):

```
conda install -c conda-forge xrviz
```

If you are using [virtualenv](#)/[pip](#):

```
pip install xrviz
```

Note that this will install with the minimum of optional requirements. If you want a more complete install, use [conda](#) instead.

If you want to develop:

```
git clone https://github.com/intake/xrviz
cd xrviz
pip install -e .
```

### 1.2 Example

To open a local or remote data in the interface:

```
import xarray as xr
from xrviz.dashboard import Dashboard

# open data with xarray
data = xr.tutorial.open_dataset('air_temperature')

# or open remote data with xarray
url = 'http://opendap.co-ops.nos.noaa.gov/thredds/dodsC/NOAA/LOOFS/MODELS/201907/
↳ glofs.loofs.fields.forecast.20190720.t06z.nc'
data = xr.open_dataset(url)

# pass the data to Dashboard
dash = Dashboard(data)
dash.show()
```

Note that `dash.show()` will open a new tab in browser, while `dash.panel` is applicable only for a jupyter cell.

You can view the example dashboard by running following in command line (this will open a tab in your browser):

```
python -c "import xrviz; xrviz.example() "
```

The above line will initialise the interface, which looks as follows:

## 1.3 Overview

The user input section of the interface constitutes of five sub-sections composed of **Panel widgets**. These sub-sections have been arranged in tabs, each of which has been described in detail in the following sections:

1. *Variables*
2. *Set\_Coords*
3. *Axes*
4. *Style*
5. *Projection*

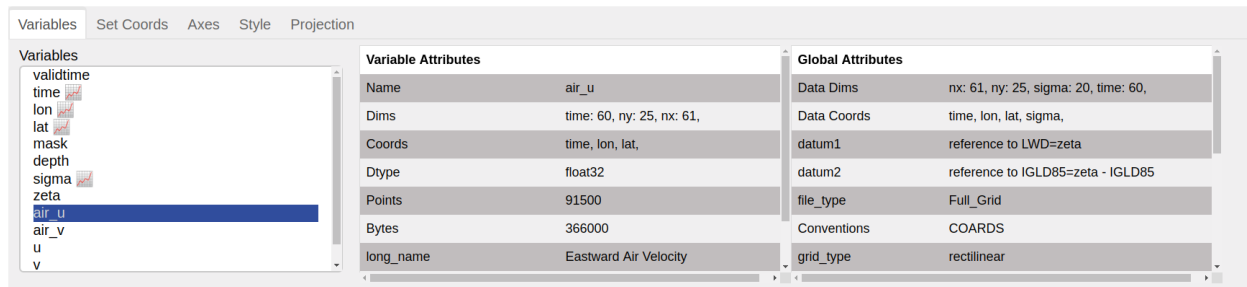
The user can interact with the widgets present in these panes to select desired inputs. Together, they govern the output produced upon pressing the `Plot` button.



## GRAPHICAL USER INTERFACE

The following five panes are part of XrViz’s interface:

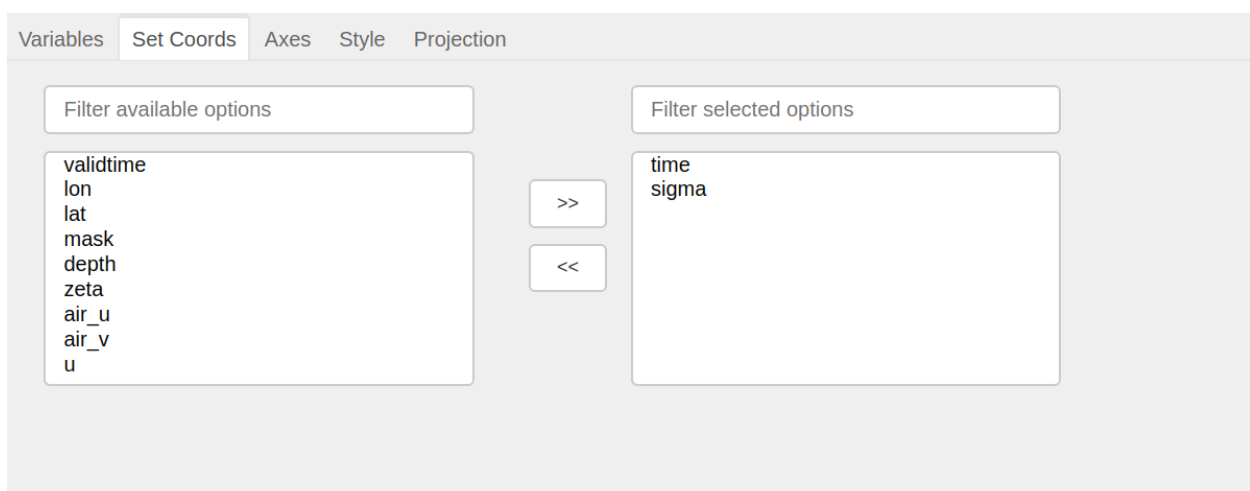
### 2.1 Variables



This pane displays [Xarray variables](#) in the list, with [coordinate variables](#) indicated by the graph icon(). Attributes for the selected variable and global attributes for the dataset are displayed on the right.

More information about this pane in [Display](#) and [Describe](#).

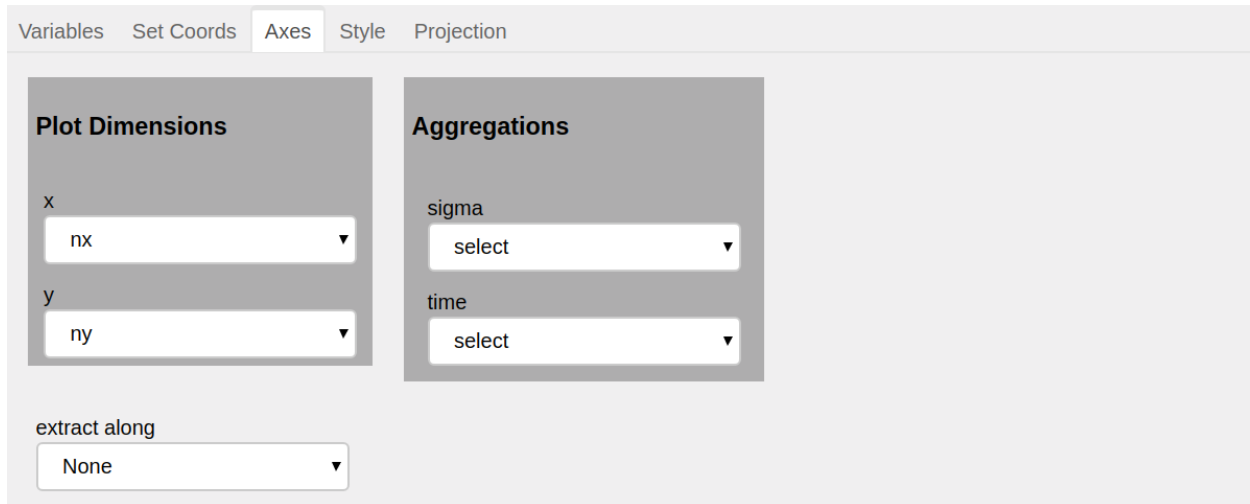
### 2.2 Set\_Coords



This pane allows the list of data coordinates to be modified by adding or removing variables using the << and >> widgets.

More information about this pane in [CoordSetter](#).

## 2.3 Axes



The screenshot shows the 'Axes' pane of the XrViz interface. It has a tabbed header with 'Variables', 'Set Coords', 'Axes' (selected), 'Style', and 'Projection'. The main area is divided into two panels: 'Plot Dimensions' and 'Aggregations'. In 'Plot Dimensions', the 'x' axis is set to 'nx' and the 'y' axis is set to 'ny'. In 'Aggregations', the 'sigma' and 'time' options are both set to 'select'. Below these panels, there is an 'extract along' dropdown menu currently set to 'None'.

This pane controls which array dimensions should be mapped, how additional dimensions should be handled, and which dimension series plots should be extracted along. The options available in this pane, update upon selection of a new variable.

It has three different sub-sections.

### 2.3.1 1. Plot Dimensions

It has following two selectors:

- **x**: To select which of the available dimensions/coordinates in the data is assigned to the plot's x (horizontal) axis.
- **y**: To select which of the available dimensions/coordinates in the data is assigned to the plot's y (vertical) axis.

**x** selector has both variable dims and coordinates available as options. However, the options available in the **y** selector depend on the selection for **x**. If the selection in **x** is a dimension, **y** will have remaining dimensions as options. Similarly, for the case when a coordinate has been selected in **x**, only remaining coordinates will be available in **y**.

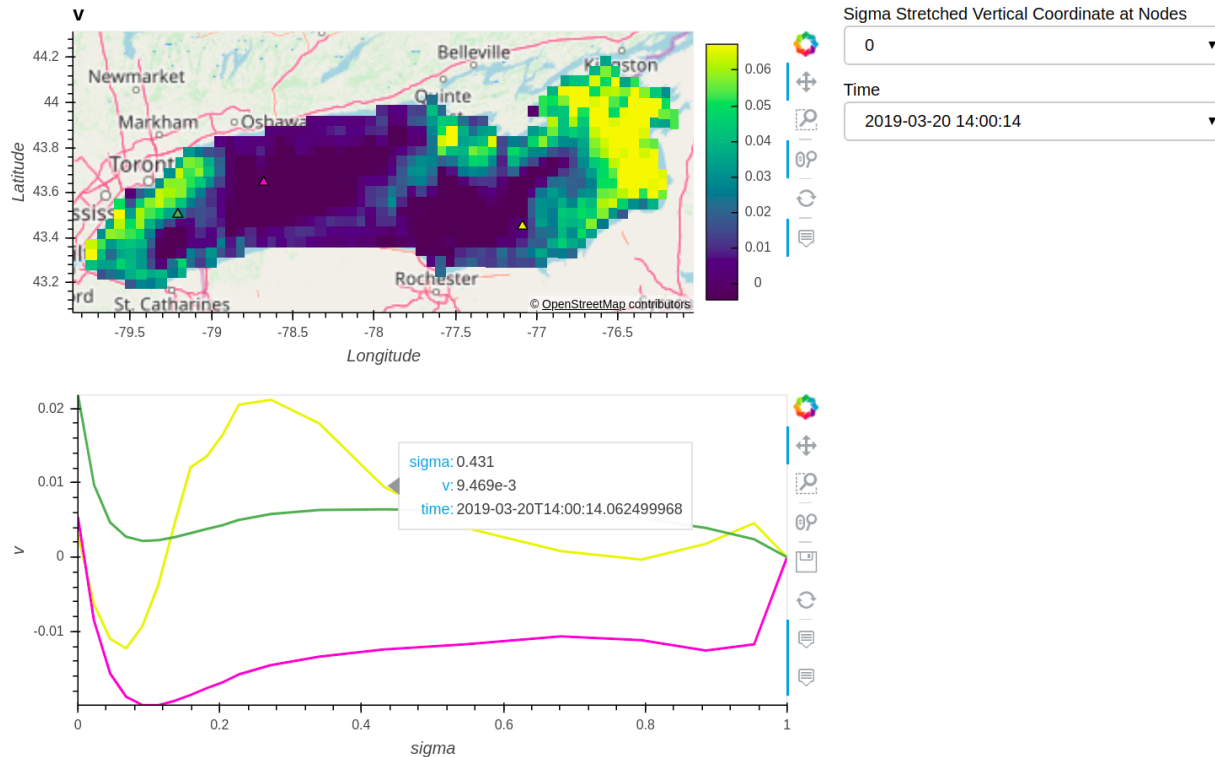
For the data following the CF conventions, the default value of **x** and **y** is filled according to guess made using [Metpy](#). In other cases it is filled according to alphabetical order of the available options.

### 2.3.2 2. Aggregations

The remaining dimensions (which have not been selected or present in coordinates selected for **x** and **y**) can be aggregated along.

### 2.3.3 3. Extract Along

This selector provides the option to select the dimension along which to create a series graph. The default option is `None`, with which this feature is disabled. After selecting a dimension and clicking `PLOT`, the user can click on the main 2D graphical output, and have a graph of the third dimension at that point appear below it. Each click will produce a marker on the main plot and a line in the series graph, which will have same colour as that of the marker. Several lines can be over-plotted. Each line in the series graph displays the value of dimensions/coordinates for which it has been extracted upon hovering over it. Also the series would be extracted in accordance to the values for which main graph has been created. Upon selecting a new dimension to extract along, the previous markers and series graph will clear.



More information about this pane in in [Axes](#).

#### Note:

1. `Clear` button is applicable only for series extraction. It clears the markers and series graph when clicked.
2. Series extraction is independent of aggregation i.e. it is possible to aggregate and extract along a same dimension.

## 2.4 Style

Variables Set Coords Axes **Style** Projection

height: 300 width: 700

cmap: Inferno color\_scale: linear

cmap lower limit: cmap upper limit:

☐ compute min/max from all data ☒ colorbar ☒ rasterize

This pane provides the options to customize the style of the output graph.

More information about this pane in [Style](#).

## 2.5 Projection

Variables Set Coords Axes Style **Projection**

☒ is\_geo

alpha: 0.70 basemap: None

crs: PlateCarree crs params: {'central\_longitude': 0.0, 'globe': None}

projection: None projection params: {}

☐ project ☐ global\_extent

features: None, borders, coastline, grid

This pane provides the option to project the data on a map or [cartopy projection](#) in case it is geographical. The geographic plots are created using [Geoviews](#), hence giving the option to visualize geographical, meteorological, and oceanographic datasets.

More information about this pane in [Projection](#).

## COMMAND LINE INTERFACE

The interface can also be launched via command line by:

```
xrviz show datafile/url
```

Here `datafile` or `url` refers to location of the file. This will open a new tab in the browser, to display the interface.



## SET INITIAL PARAMETERS

Users may need to initialize the widgets present in dashboard with custom values instead of the defaults. This could be done by passing initial parameters to the *Dashboard* upon initialization. All the parameters are passed using the argument `initial_params` which is of type `dict`.

The keys present in `initial_params` must match with the name of the widgets, whose values you want to change. Also the value passed must be present in the available options of that widget. All the keys must be of type `str`.

Example:

```
from .sample_data import great_lakes
initial_params = {# Select Variable
                  'Variables': 'temp',

                  # Set Coords
                  'Set Coords': ['lat', 'lon'],

                  # Axes
                  'x': 'lon',
                  'y': 'lat',
                  'sigma': 'animate',

                  # Style
                  'height': 300,
                  'width': 650,
                  'colorbar': True,
                  'cmap': 'Viridis',

                  # Projection
                  'is_geo': True,
                  'basemap': 'OSM',
                  'crs': 'PlateCarree',
                  'projection': 'Orthographic',
                  'crs params': '{"central_longitude': 0.0}",
                  'projection params': '{"central_longitude': -78, 'central_latitude
→': 43, 'globe': {'ellipse': 'sphere'}}"
                  }
dash = Dashboard(great_lakes, initial_params=initial_params)
dash.show()
```

Here `crs params` and `projection params` need more attention. The parameter passed to these must be accepted by the selected projection as defined in [cartopy projections](#).





## 5.1 Variables

**class** `xrviz.display.Display` (*data*)

Displays a list of data variables for selection.

For each data, its variables are displayed in `pn.widgets.MultiSelect`. Variables which are data coordinates are annotated with “.”.

### Parameters

**data:** `xarray.DataSet`

### Attributes

**panel:** A `panel.Row` instance which displays the Multiselect widget.

### Methods

---

<code>select_variable</code> ( <i>self</i> , <i>variable</i> )	Select a data variable from the available options.
--	--

---

**select\_variable** (*self*, *variable*)

Select a data variable from the available options.

**class** `xrviz.describe.Describe` (*data*)

Displays the properties of the variable selected in the Display section.

This section has two tables as output. The first table shows *Variable attributes* and the second table shows *Global Attributes*. Upon selection of a new variable in the Display widget, the first table updates itself with properties of the new selection, while the second table stays same.

### Parameters

**data:** `xarray.DataSet`

### Attributes

**panel:** A `panel.pane.HTML` instance which displays the tables arranged next to each other.

### Methods

---

<code>variable_pane_for_dataset(self, var)</code>	Returns HTML template describing variable and global attributes.
---	--

---

**variable\_pane\_for\_dataset** (*self*, *var*)  
Returns HTML template describing variable and global attributes.

## 5.2 Set\_Coords

**class** `xrviz.coord_setter.CoordSetter` (*data*)

An input pane for choosing which variables are considered coordinates.

It uses a [Cross Selector](#) to display a list of simple and coordinate variables. Simple variables (which are not data coordinates) are available on left side and default coordinates are available on right side. To set variables as coordinates, make selection on left side and click >>. Similarly making selection on right side and clicking << will reset the coordinates. Other panes update themselves accordingly, in response to this change.

### Methods

---

<code>set_coords(self, data)</code>	Called when the data attribute of the interface has change in variables considered as coordinates.
<code>setup_initial_values(self[, init_params])</code>	To set the variables, whose names have been passed, as coordinates.

---

**set\_coords** (*self*, *data*)  
Called when the data attribute of the interface has change in variables considered as coordinates.

**setup\_initial\_values** (*self*, *init\_params*={})  
To set the variables, whose names have been passed, as coordinates.

## 5.3 Axes

**class** `xrviz.fields.Fields` (*data*)

To select the fields to plot along.

This pane controls which array dimensions should be mapped, how additional dimensions should be handled, and which dimension series plots should be extracted along.

### Parameters

**data:** `xarray.DataSet`

### Attributes

**x:** To select which of the available dimensions/coordinates in the data is assigned to the plot's x (horizontal) axis.

**y:** To select which of the available dimensions/coordinates in the data is assigned to the plot's y (vertical) axis.

**Remaining Dims:** Any one of the following aggregations can be applied on each of remaining dimensions:

1. `select`: It creates a `pn.widgets.Select`, to select the value of dimension, for which the graph would be displayed.
2. `animate`: It creates a `panel.widgets.DiscretePlayer` which helps to quickly iterate over all the values for a dimension.
3. `mean`: Creates plot along mean of the selected dimension.
4. `max`: Creates plot along maximum of the selected dimension.
5. `min`: Creates plot along minimum of the selected dimension.
6. `median`: Creates plot along median of the selected dimension.
7. `std`: Creates plot along standard deviation of the selected dimension.
8. `count`: Creates plot along non-nan values of the selected dimension.

Note that for both `select` and `animate`, the plot will update according to the value selected in the generated widget. Also, if a dimension has been aggregated, its select widget would not be available.

**Extract Along:** This selector provides the option to select the dimension along which to create a series graph.

## Methods

<code>change_dim_selectors(self, *args)</code>	Updates the dimensions available for <i>Aggregation</i> and <i>Extract Along</i> upon change in value of <code>y</code> .
<code>change_y(self[, value])</code>	Updates <code>y</code> by removing the value of <code>x</code> , from the available options
<code>check_are_var_coords(self)</code>	Check if both <code>x</code> and <code>y</code> are coordinates of the selected variable.
<code>guess_x_y(self, var)</code>	To guess the value of <code>x</code> and <code>y</code> with <code>metpy.parse_cf</code> .
<code>setup(self, var)</code>	Fill available options for the selected variable.

**change\_dim\_selectors** (*self*, \*args)

Updates the dimensions available for *Aggregation* and *Extract Along* upon change in value of `y`.

**change\_y** (*self*, value=None)

Updates `y` by removing the value of `x`, from the available options

**check\_are\_var\_coords** (*self*)

Check if both `x` and `y` are coordinates of the selected variable.

**guess\_x\_y** (*self*, var)

To guess the value of `x` and `y` with `metpy.parse_cf`. This is applicable only for the case when both `x` and `y` are data coordinates.

**setup** (*self*, var)

Fill available options for the selected variable.

## 5.4 Style

**class** `xrviz.style.Style`

A pane to customise styling of the graphs.

The following options are available in this pane:

1. **height (default 300):** To modify the height of main and series graph.
2. **width (default 700):** To modify the width of the main and series graph.
3. **cmap (default *Inferno*):** To select a colormap for the main graph.
4. **color\_scale (default *linear*):** To scale the values to be plotted. The scaling options available are `linear`, `exp`, `log`, `reciprocal`, `square` and `sqrt`. Here `linear` implies no scaling.
5. **cmap limits:** To change the colormap limits. User can fill these limits before plotting a variable. In case not filled by user, automatic filling of limits happen.

- `lower limit`: auto-filled value equals `quantile(0.1)` of values to be plotted.
- `upper limit`: auto-filled value equals `quantile(0.9)` of values to be plotted.

In case of dask array, `dask.array.percentile` is use to compute the limits. `tdigest` method is used in case `crick` is present. The value of limits is rounded off to 5 decimal places, for simplicity.

Note that these values are filled with respect to color scaled values. Also these limits clear upon change in variable or color scaling.

6. **compute min/max from all data (default *False*):**
  - `True`: all values present in a data variable are used to compute upper and lower colormap limits.
  - `False`: only values necessary to create first step/instance of graph are used.

It is better to have its value *False* for larger datasets, to save computation time.

7. **colorbar (default *True*):** Provides option to display/hide colorbar.
8. **rasterize (default *True*):** Provides option to use `data shading` . It is better to have its value *True*, to get highly optimized rendering.

## Methods

---

<code>setup_initial_values(self[, init_params])</code>	To select initial values for the widgets in this pane.
--	--

---

**setup\_initial\_values** (*self*, *init\_params*={})  
To select initial values for the widgets in this pane.

## 5.5 Projection

## 5.6 Control

**class** `xrviz.control.Control` (*data*)

The user input part of the interface

### Parameters

**data:** `xarray.DataSet` The data to be visualised. Here, we are mostly concerned with displaying the variables, their attributes, and assigning coordinates to roles upon plotting.

### Attributes

1. **panel:** A `panel.Tabs` instance containing the user input panes

2. **displayer:** A `Display` instance, displays a list of data variables for selection.
3. **describer:** A `Describe` instance, describes the properties of the variable selected in the `displayer`.
4. **coord\_setter:** A `CoordSetter` instance for choosing which variables are considered coordinates.
5. **fields:** A `Fields` instance to select the axes to plot with.
6. **style:** A `Style` instance to customise styling of the graphs.
7. **projection:** A `Projection` instance to customise the projection of geographical data.
8. **kwargs:** A dictionary gathered from the widgets of the input Panes, of a form which can be passed to the plotting function as kwargs.

## Methods

<code>check_is_projectable(self, \*args)</code>	Check if the selected variable can be projected geographically.
<code>set_coords(self, data)</code>	Called after coordinates have been set, to update the other input panes.

**check\_is\_projectable** (*self*, \*args)

Check if the selected variable can be projected geographically.

This is possible only when both *x* and *y* are present in selected variable's coordinates.

**set\_coords** (*self*, data)

Called after coordinates have been set, to update the other input panes.

## 5.7 Dashboard

**class** `xrviz.dashboard.Dashboard` (data, initial\_params={})

Main entry point to XrViz, an interactive GUI for a given dataset.

### Parameters

**data:** `xarray.DataSet` The data to be visualised

**initial\_params:** 'dict' To pre-select values of widgets upon initialization. The keys are generally names of widgets within the input area of the interface. For more details, refer to [Set Initial Parameters](#).

### Attributes

1. **panel:** A `panel.Tabs` instance containing the user input panes and output graphs of the interface.
2. **control:** A `Control` instance responsible for input panes (control panel).
3. **plot\_button:** A `pn.widgets.Button` that generates graph according to values selected in input panes, upon click.
4. **graph:** A `HoloViews(DynamicMap)` instance containing the main graph.
5. **output:** The graph along with the select widgets for index selection.

- 6. **taps\_graph**: A `holoviews.Points` instance to record the location of taps.
- 7. **series\_graph**: A `HoloViews(Overlay)` instance having series extracted.
- 8. **clear\_series\_button**: A `pn.widgets.Button` to clear the *taps\_graph* and *series\_graph*.

## Methods

<code>check_is_plottable(self, var)</code>	Check if a data variable can be plotted.
<code>clear_series(self, \*args)</code>	Clears the markers on the image, and the extracted series.
<code>create_graph(self, \*args)</code>	Creates a graph according to the values selected in the widgets.
<code>create_indexed_graph(self, \*args)</code>	Creates a graph for the dimensions selected in widgets <i>x</i> and <i>y</i> .
<code>create_selectors_players(self, graph)</code>	Converts the sliders generated by hvplot into selectors/players.
<code>create_series_graph(self, x, y, color[, clear])</code>	Extract a series at a given point, and plot it.
<code>create_taps_graph(self, x, y[, clear])</code>	Create an output layer in the graph which responds to taps

### **check\_is\_plottable** (*self*, *var*)

Check if a data variable can be plotted.

If a variable is 1-d, disable `plot_button` for it.

### **clear\_series** (*self*, *\\*args*)

Clears the markers on the image, and the extracted series.

### **create\_graph** (*self*, *\\*args*)

Creates a graph according to the values selected in the widgets.

This method is usually invoked by the user clicking “Plot”

It handles the following two cases:

1. Both *x*, *y* are present in selected variable’s coordinates. Geographic projection is possible only in this case. It uses `create_selectors_players` method for creation of the graph. Here the selectors generated automatically by hvplot are used.
2. One or both of *x*, *y* are NOT present in selected variable’s coordinates (both *x* and *y* are considered as dimensions). It uses `create_indexed_graph` method for creation of the graph. The selectors are created and linked with graph by XrViz.

### **create\_indexed\_graph** (*self*, *\\*args*)

Creates a graph for the dimensions selected in widgets *x* and *y*.

This is used when values selected in *x* and *y* are not data coordinates (i.e. one or both values are data dimensions).

### **create\_selectors\_players** (*self*, *graph*)

Converts the sliders generated by hvplot into selectors/players.

This is applicable only when both *x* and *y* are present in variable coordinates. It converts any sliders generated by hvplot into selectors/players and moves them to the bottom of graph.

### **create\_series\_graph** (*self*, *x*, *y*, *color*, *clear=False*)

Extract a series at a given point, and plot it.

The series plotted has same color as that of the marker depicting the location of the tap.

**The following cases have been handled:**

**Case 1:** When both x and y are NOT coords (i.e. are dims)

**Case 2:** When both x and y are coords

2a: Both are 1-dimensional

2b: Both are 2-dimensional with same dimensions.

2c: Both are 2-dimensional with different dims or are multi-dimcoordinates. Here we are unable to extract.

Note that Case 1 and Case 2a can be handled with the same code.

**create\_taps\_graph** (*self*, *x*, *y*, *clear=False*)

Create an output layer in the graph which responds to taps

Whenever the user taps (or clicks) the graph, a glyph will be overlaid, and a series is extracted at that point.

## 5.8 Example

`xrviz.example` (*show=True*)

Generates interface for the sample dataset.

### Parameters

**show: bool (True)** True: A new browser tab will be opened, and the function will block until interrupted. False: The Dashboard instance is returned without being executed.





## INDICES AND TABLES

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



## C

`change_dim_selectors()` (*xrviz.fields.Fields method*), 15  
`change_y()` (*xrviz.fields.Fields method*), 15  
`check_are_var_coords()` (*xrviz.fields.Fields method*), 15  
`check_is_plottable()` (*xrviz.dashboard.Dashboard method*), 18  
`check_is_projectable()` (*xrviz.control.Control method*), 17  
`clear_series()` (*xrviz.dashboard.Dashboard method*), 18  
`Control` (*class in xrviz.control*), 16  
`CoordSetter` (*class in xrviz.coord\_setter*), 14  
`create_graph()` (*xrviz.dashboard.Dashboard method*), 18  
`create_indexed_graph()` (*xrviz.dashboard.Dashboard method*), 18  
`create_selectors_players()` (*xrviz.dashboard.Dashboard method*), 18  
`create_series_graph()` (*xrviz.dashboard.Dashboard method*), 18  
`create_taps_graph()` (*xrviz.dashboard.Dashboard method*), 19

## D

`Dashboard` (*class in xrviz.dashboard*), 17  
`Describe` (*class in xrviz.describe*), 13  
`Display` (*class in xrviz.display*), 13

## E

`example()` (*in module xrviz*), 19

## F

`Fields` (*class in xrviz.fields*), 14

## G

`guess_x_y()` (*xrviz.fields.Fields method*), 15

## S

`select_variable()` (*xrviz.display.Display method*),  
 13

`set_coords()` (*xrviz.control.Control method*), 17  
`set_coords()` (*xrviz.coord\_setter.CoordSetter method*), 14  
`setup()` (*xrviz.fields.Fields method*), 15  
`setup_initial_values()` (*xrviz.coord\_setter.CoordSetter method*), 14  
`setup_initial_values()` (*xrviz.style.Style method*), 16  
`Style` (*class in xrviz.style*), 15

## V

`variable_pane_for_dataset()` (*xrviz.describe.Describe method*), 14