
anesthetic Documentation

Release 2.0.0-beta.1

Will Handley

Jul 31, 2020

Contents

1 Features	3
2 Installation	5
3 Documentation	7
4 Citation	9
5 Contributing	11
6 Questions/Comments	13
7 anesthetic package	15
8 anesthetic: nested sampling visualisation	63
Python Module Index	69
Index	71

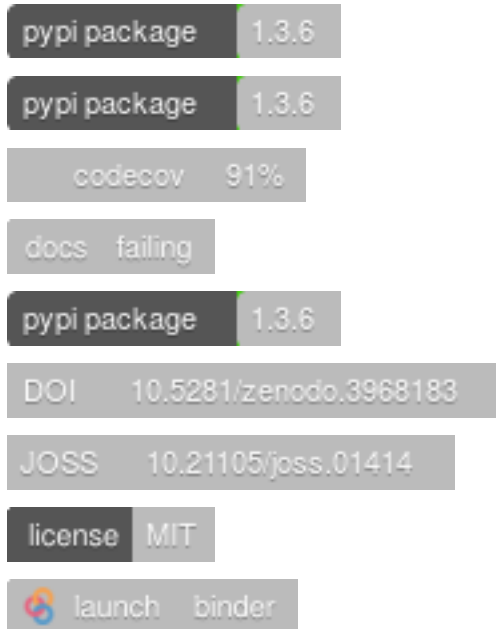
anesthetic nested sampling visualisation

Author Will Handley and Lukas Hergt

Version 2.0.0-beta.1

Homepage <https://github.com/williamjameshandley/anesthetic>

Documentation <http://anesthetic.readthedocs.io/>



`anesthetic` brings together tools for processing nested sampling chains, leveraging standard scientific python libraries.

You can see example usage and plots in the [plot gallery](#), or in the corresponding [Jupyter notebook](#).

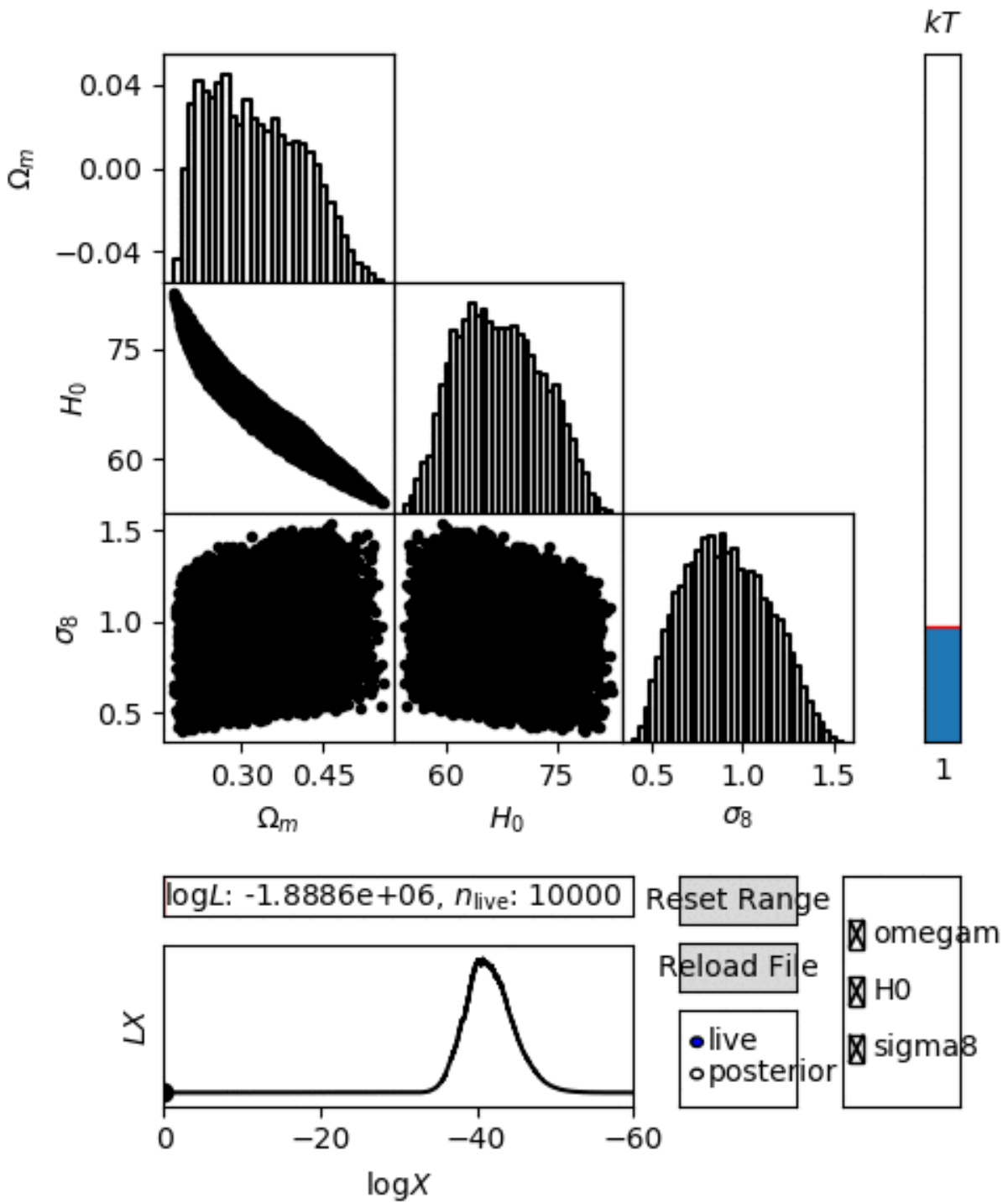
Current functionality includes:

- Computation of Bayesian evidences, Kullback-Liebler divergences and Bayesian model dimensionalities.
- Marginalised 1d and 2d plots.
- Dynamic replaying of nested sampling.

This tool was designed primarily for use with nested sampling outputs, although it can be used for normal MCMC chains.

For an interactive view of a nested sampling run, you can use the `anesthetic` script.

```
$ anesthetic <ns file root>
```



CHAPTER 1

Features

- Both samples and plotting axes are stored as a `pandas.DataFrame`, with parameter names as indices, which makes for easy access and modification.
- Sensible color scheme for plotting nearly flat distributions.
- For easy extension/modification, uses the standard python libraries: `numpy`, `scipy`, `matplotlib` and `pandas`.

CHAPTER 2

Installation

anesthetic can be installed via pip

```
pip install anesthetic
```

or via the setup.py

```
git clone https://github.com/williamjameshandley/anesthetic
cd anesthetic
python setup.py install --user
```

You can check that things are working by running the test suite:

```
export MPLBACKEND=Agg      # only necessary for OSX users
python -m pytest
flake8 anesthetic tests
pydocstyle --convention=numpy anesthetic
```

2.1 Dependencies

Basic requirements:

- Python 3.6+
- matplotlib
- numpy
- scipy
- pandas
- fastKDE

Documentation:

- sphinx
- numpydoc

Tests:

- pytest

CHAPTER 3

Documentation

Full Documentation is hosted at [ReadTheDocs](#). To build your own local copy of the documentation you'll need to install [sphinx](#). You can then run:

```
cd docs
make html
```


If you use `anesthetic` to generate plots for a publication, please cite as:

```
Handley, (2019). anesthetic: nested sampling visualisation. Journal of Open Source Software, 4(37), 1414, https://doi.org/10.21105/joss.01414
```

or using the BibTeX:

```
@article{anesthetic,  
  doi = {10.21105/joss.01414},  
  url = {http://dx.doi.org/10.21105/joss.01414},  
  year = {2019},  
  month = {Jun},  
  publisher = {The Open Journal},  
  volume = {4},  
  number = {37},  
  pages = {1414},  
  author = {Will Handley},  
  title = {anesthetic: nested sampling visualisation},  
  journal = {The Journal of Open Source Software}  
}
```


CHAPTER 5

Contributing

There are many ways you can contribute via the [GitHub repository](#).

- You can [open an issue](#) to report bugs or to propose new features.
- Pull requests are very welcome. Note that if you are going to propose major changes, be sure to open an issue for discussion first, to make sure that your PR will be accepted before you spend effort coding it.

6.1 Another posterior plotting tool?

This is my posterior plotter. There are many like it, but this one is mine.

There are several excellent tools for plotting marginalised posteriors:

- `getdist`
- `corner`
- `pygtc`
- `dynesty`
- `MontePython`

Why create another one? In general, any dedicated user of software will find that there is some functionality that in their use case is lacking, and the designs of previous codes make such extensions challenging. In my case this was:

1. For large numbers of samples, kernel density estimation is slow, or inaccurate (particularly for samples generated from nested sampling). There are kernel density estimators, such as `fastKDE`, which ameliorate many of these difficulties.
2. Existing tools can make it difficult to define new parameters. For example, the default `cosmomic` chain defines `omegabh2`, but not `omegab`. The transformation is easy, since $\text{omegab} = \text{omegab}_{h2} / (H_0/100)^{**2}$, but implementing this transformation in existing packages is not so trivial. `anesthetic` solves this issue by storing the samples as a pandas array, for which the relevant code for defining the above new parameter would be

```
from anesthetic import MCMCSamples
samples = MCMCSamples(root=file_root)           # Load the samples
samples['omegab'] = samples.omegabh2 / (samples.H0/100)**2  # Define omegab
samples.tex['omegab'] = '$\Omega_b$'           # Label omegab
samples.plot_1d('omegab')                     # Simple 1D plot
```

3. Many KDE plotting tools have conventions that don't play well with uniformly distributed parameters, which presents a problem if you are trying to plot priors along with your posteriors. `anesthetic` has a sensible mechanism, by defining the contours by the amount of iso-probability mass they contain, but colouring the fill in relation to the probability density of the contour.

6.2 What's in a name?

There is an emerging convention for naming nested sampling packages with words that have nest in them (`nestle` and `dynesty`, `nestorflow`). Doing a UNIX `grep`:

```
grep nest /usr/share/dict/words
```

yields a lot of superlatives (e.g. `greenest`), but a few other cool names for future projects:

- `amnesty`
- `defenestrate`
- `dishonestly`
- `inestimable`
- `minestrone`
- `rhinestone`

I chose `anesthetic` because I liked the soft 'th', and in spite of the US spelling.

7.1 Subpackages

7.1.1 anesthetic.gui package

Submodules

anesthetic.gui.plot module

Main plotting tools.

class `anesthetic.gui.plot.Evolution` (*fig, gridspec, action, valmax*)
 Bases: `anesthetic.gui.widgets.Slider`

Slider controlling the evolution stage of the live points.

Methods

<code>__call__()</code>	Return the current iteration as an integer.
<code>reset_range([valmin, valmax])</code>	Reset the range of the slider.
<code>set_text(logL, n)</code>	Set the text at end of slider.

set_text (*logL, n*)
 Set the text at end of slider.

Parameters

logL: float

Current loglikelihood of evolution stage

n: int Current number of live points of evolution stage

class `anesthetic.gui.plot.Higson` (*fig, gridspec*)

Bases: `anesthetic.gui.widgets.Widget`

Higson plot as shown in <https://arxiv.org/abs/1703.09701>.

Attributes

curve: `matplotlib.lines.Line2D`

points currently plotted as a curve.

point: `matplotlib.lines.Line2D` large indicator point currently plotted on the curve.

Methods

<code>reset_range()</code>	Reset the ranges of the higson plot.
<code>update(logX, LX, i)</code>	Update the line and the point in the higson plot.

reset_range ()

Reset the ranges of the higson plot.

update (*logX, LX, i*)

Update the line and the point in the higson plot.

Parameters

logX: array-like

log-volume compression values to plot

LX: array-like Likelihood * volume compression

i: int Current location of higson point

class `anesthetic.gui.plot.RunPlotter` (*samples, params=None*)

Bases: `object`

Construct a control panel of information on a nested sampling run.

Parameters

samples: `anesthetic.samples.NestedSamples` The root string for the chains files to be used, or a set of nested samples.

Attributes

samples: `anesthetic.samples.NestedSamples`

Object for extracting nested sampling data from chains files.

fig: `matplotlib.figure.Figure` Reference to the underlying figure

triangle: `anesthetic.gui.widgets.TrianglePlot` Corner plot of live or posterior samples.

temperature: `anesthetic.gui.plot.Temperature` Slider selecting the posterior temperature.

evolution: `anesthetic.gui.plot.Evolution` Slider selecting the live iteration.

higson: `anesthetic.gui.plot.Higson` Higson plot of posterior weights.

reset: `anesthetic.gui.widgets.Button` Button that resets the parameter ranges.

reload: `anesthetic.gui.widgets.Button` Button that reloads the files.

type: `anesthetic.gui.widgets.RadioButton`s Radio buttons that selects whether to plot live or posteriors.

param_choice: `anesthetic.gui.widgets.CheckButtons` Checkbox that selects which parameters to plot.

Methods

<code>points(label)</code>	Get sample coordinates from nested sampling samples.
<code>redraw(_)</code>	Redraw the triangle plot upon parameter updating.
<code>reload_file(_)</code>	Reload the data from file.
<code>reset_range(_)</code>	Reset the parameter ranges.
<code>update(_)</code>	Update all the plots upon slider changes.

`points` (*label*)

Get sample coordinates from nested sampling samples.

Parameters

label: `str` label indicating the coordinate to extract.

Returns

array-like: sample 'label'-coordinates.

`redraw` (*_*)

Redraw the triangle plot upon parameter updating.

`reload_file` (*_*)

Reload the data from file.

`reset_range` (*_*)

Reset the parameter ranges.

`update` (*_*)

Update all the plots upon slider changes.

class `anesthetic.gui.plot.Temperature` (*fig, gridspec, action*)

Bases: `anesthetic.gui.widgets.Slider`

Logarithmic slider controlling temperature of the posterior points.

Methods

<code>__call__()</code>	Return the current temperature.
<code>reset_range([valmin, valmax])</code>	Reset the range of the slider.
<code>set_text(kT)</code>	Set the text at end of slider.

`set_text` (*kT*)

Set the text at end of slider.

Parameters

kT: `float` Current temperature of posterior points stage

anesthetic.gui.widgets module

Widget wrappers to matplotlib.

These extend the matplotlib widgets by plotting themselves onto an axis and storing a reference to both the widget object and the axis on which they are plotted.

class `anesthetic.gui.widgets.Button` (*fig, gridspec, action, text*)
 Bases: `anesthetic.gui.widgets.Widget`

Push button that performs an action.

Parameters

action: func

What should be run upon clicking the button.

text: str Overlay text on the button.

Attributes

button: matplotlib.widgets.Button matplotlib Button.

class `anesthetic.gui.widgets.CheckButtons` (*fig, gridspec, labels, action*)
 Bases: `anesthetic.gui.widgets.LabelsWidget`

Set of checkboxes.

Defaults to choosing the only the first label at start.

Parameters

action: func What should be done upon checking the box.

Attributes

buttons: matplotlib.widgets.CheckButtons matplotlib CheckButtons.

Methods

<code>__call__()</code>	Get current status of the check boxes.
-------------------------	----------------------------------------

class `anesthetic.gui.widgets.LabelsWidget` (*fig, gridspec, labels*)
 Bases: `anesthetic.gui.widgets.Widget`

Widget with labels to choose from.

Parameters

labels: list(str) Set of labels to be tied to Widget.

Attributes

labels: list(str) Set of labels on Widget.

class `anesthetic.gui.widgets.RadioButton`s (*fig, gridspec, labels, action*)
 Bases: `anesthetic.gui.widgets.LabelsWidget`

Set of radio selection choices.

Parameters

labels: list(str)

Attributes

buttons: `matplotlib.widgets.RadioButtons` matplotlib RadioButtons.

Methods

<code>__call__()</code>	Get current selection string.
-------------------------	-------------------------------

class `anesthetic.gui.widgets.Slider` (*fig, gridspec, action, text, valmin, valmax, valinit, orientation*)

Bases: `anesthetic.gui.widgets.Widget`

Choose a parameter via a slider widget.

Parameters

action: `func`

What should be done upon altering the slider.

text: `str` Text at the start of the slider.

valmin, valmax, valinit: `float` Range and initial value for the slider.

orientation: `str` Orientation for slider ('horizontal' or 'vertical').

Attributes

slider: `matplotlib.widgets.Slider` matplotlib Slider.

Methods

<code>__call__()</code>	Return the float value in the slider.
<code>reset_range([valmin, valmax])</code>	Reset the range of the slider.
<code>set_text(text)</code>	Set the text at the end of the slider.

reset_range (*valmin=None, valmax=None*)
Reset the range of the slider.

Kwargs:

valmin, valmax: (`float`), **optional:** The new limits to the slider.

set_text (*text*)
Set the text at the end of the slider.

Parameters

text: `str` Text to be chosen

class `anesthetic.gui.widgets.TrianglePlot` (*fig, gridspec*)

Bases: `anesthetic.gui.widgets.Widget`

Triangle plot widget as exemplified by `getdist` and `corner.py`.

For other examples of these plots, see: <https://getdist.readthedocs.io> <https://corner.readthedocs.io>

Attributes

ax: `pandas.DataFrame(matplotlib.axes.Axes)` Mapping from pairs of parameters to axes for plotting.

Methods

<code>draw(labels[, tex])</code>	Draw a new triangular grid for list of parameters labels.
<code>reset_range()</code>	Reset the range of each grid.
<code>update(f)</code>	Update the points in the triangle plot using f function.

draw (*labels*, *tex*={})

Draw a new triangular grid for list of parameters labels.

Parameters

labels: `list(str)` labels for the triangular grid.

reset_range ()

Reset the range of each grid.

update (*f*)

Update the points in the triangle plot using f function.

Parameters

f: `func: str -> array(float)` this function should take in a parameter label *i*, and return an array-like object of the *i*-coordinate of the samples

class `anesthetic.gui.widgets.Widget` (*fig*, *gridspec*)

Bases: `object`

Base class for anesthetic gui widgets.

Stores a reference to the underlying figure, the gridspec that the widget is placed at and the axes of the widget.

Parameters

fig: `matplotlib.figure.Figure`

Figure for drawing widget on.

gridspec: `matplotlib.gridspec.GridSpec` Specification for where to draw in the figure. Technically could be any argument that can be passed to `matplotlib.figure.Figure.add_subplot`.

Attributes

fig: (`matplotlib.figure.Figure`)

Figure for drawing widget on.

gridspec: `matplotlib.gridspec.GridSpec` Specification for where to draw in the figure. Technically could be any argument that can be passed to `matplotlib.figure.Figure.add_subplot`.

ax: `matplotlib.axes.Axes` Axes of widget.

Module contents

Graphical user interface for inspecting nested sampling runs.

7.1.2 anesthetic.read package

Submodules

anesthetic.read.base module

anesthetic.read.getdist module

anesthetic.read.multinest module

anesthetic.read.nested module

anesthetic.read.polychord module

Module contents

Module for reading samples from file.

7.2 Submodules

7.3 anesthetic.kde module

7.4 anesthetic.plot module

Lower-level plotting tools.

Routines that may be of use to users wishing for more fine-grained control may wish to use.

- `make_1d_axes`
- `make_2d_axes`
- `get_legend_proxy`

to create a set of axes and legend proxies.

`anesthetic.plot.basic_cmap` (*color*)

Construct basic colormap a single color.

`anesthetic.plot.fastkde_contour_plot_2d` (*ax*, *data_x*, *data_y*, **args*, ***kwargs*)

Plot a 2d marginalised distribution as contours.

This functions as a wrapper around `matplotlib.axes.Axes.contour`, and `matplotlib.axes.Axes.contourf` with a kernel density estimation computation in between. All remaining keyword arguments are passed onwards to both functions.

Parameters

ax: `matplotlib.axes.Axes` axis object to plot on

data_x, data_y: np.array x and y coordinates of uniformly weighted samples to generate kernel density estimator.

levels: list amount of mass within each iso-probability contour. optional, default [0.68, 0.95]

xmin, xmax, ymin, ymax: float lower/upper prior bounds in x/y coordinates optional, default None

Returns

c: matplotlib.contour.QuadContourSet A set of contourlines or filled regions

`anesthetic.plot.fastkde_plot_1d(ax, data, *args, **kwargs)`

Plot a 1d marginalised distribution.

This functions as a wrapper around `matplotlib.axes.Axes.plot`, with a kernel density estimation computation provided by the package `fastkde` in between. All remaining keyword arguments are passed onwards.

Parameters

ax: matplotlib.axes.Axes axis object to plot on

data: np.array Uniformly weighted samples to generate kernel density estimator.

xmin, xmax: float lower/upper prior bound optional, default None

Returns

lines: matplotlib.lines.Line2D A list of line objects representing the plotted data (same as `matplotlib matplotlib.axes.Axes.plot` command)

`anesthetic.plot.get_legend_proxy(fig)`

Extract a proxy for plotting onto a legend.

Example usage:

```
>>> fig, axes = modelA.plot_2d()
>>> modelB.plot_2d(axes)
>>> proxy = get_legend_proxy(fig)
>>> fig.legend(proxy, ['A', 'B'])
```

Parameters

fig: matplotlib.figure.Figure Figure to extract colors from.

`anesthetic.plot.hist_plot_1d(ax, data, *args, **kwargs)`

Plot a 1d histogram.

This functions is a wrapper around `matplotlib.axes.Axes.hist`. All remaining keyword arguments are passed onwards.

Parameters

ax: matplotlib.axes.Axes axis object to plot on

data: np.array Samples to generate histogram from

weights: np.array, optional Sample weights.

xmin, xmax: float lower/upper prior bound. optional, default `data.min()` and `data.max()` cannot be None (reverts to default in that case)

Returns

patches [list or list of lists] Silent list of individual patches used to create the histogram or list of such list if multiple input datasets.

Other Parameters

****kwargs** [*~matplotlib.axes.Axes.hist* properties]

`anesthetic.plot.hist_plot_2d(ax, data_x, data_y, *args, **kwargs)`

Plot a 2d marginalised distribution as a histogram.

This functions as a wrapper around `matplotlib.axes.Axes.hist2d`

Parameters

ax: `matplotlib.axes.Axes` axis object to plot on

data_x, data_y: `np.array` x and y coordinates of uniformly weighted samples to generate kernel density estimator.

xmin, xmax, ymin, ymax: `float` lower/upper prior bounds in x/y coordinates optional, default None

levels: `list` Shade iso-probability contours containing these levels of probability mass. If None defaults to usual `matplotlib.axes.Axes.hist2d` colouring. optional, default None

Returns

c: `matplotlib.collections.QuadMesh` A set of colors

`anesthetic.plot.kde_contour_plot_2d(ax, data_x, data_y, *args, **kwargs)`

Plot a 2d marginalised distribution as contours.

This functions as a wrapper around `matplotlib.axes.Axes.tricontour`, and `matplotlib.axes.Axes.tricontourf` with a kernel density estimation computation provided by `scipy.stats.gaussian_kde` in between. All remaining keyword arguments are passed onwards to both functions.

Parameters

ax: `matplotlib.axes.Axes` axis object to plot on.

data_x, data_y: `np.array` x and y coordinates of uniformly weighted samples to generate kernel density estimator.

weights: `np.array, optional` Sample weights.

ncompress: `int, optional` Degree of compression. optional, Default 1000

xmin, xmax, ymin, ymax: `float` lower/upper prior bounds in x/y coordinates. optional, default None

Returns

c: `matplotlib.contour.QuadContourSet` A set of contourlines or filled regions

`anesthetic.plot.kde_plot_1d(ax, data, *args, **kwargs)`

Plot a 1d marginalised distribution.

This functions as a wrapper around `matplotlib.axes.Axes.plot`, with a kernel density estimation computation provided by `scipy.stats.gaussian_kde` in between. All remaining keyword arguments are passed onwards.

Parameters

ax: `matplotlib.axes.Axes` axis object to plot on.

data: `np.array` Samples to generate kernel density estimator.

weights: `np.array, optional` Sample weights.

ncompress: int, optional Degree of compression. Default 1000

xmin, xmax: float lower/upper prior bound. optional, default None

Returns

lines: matplotlib.lines.Line2D A list of line objects representing the plotted data (same as matplotlib.axes.Axes.plot command)

`anesthetic.plot.make_1d_axes` (*params*, ***kwargs*)
 Create a set of axes for plotting 1D marginalised posteriors.

Parameters

params: list(str)

names of parameters.

tex: dict(str:str), optional Dictionary mapping params to tex plot labels.

fig: matplotlib.figure.Figure, optional Figure to plot on. Default: matplotlib.pyplot.figure()

ncols: int Number of columns in the plot option, default ceil(sqrt(num_params))

subplot_spec: matplotlib.gridspec.GridSpec, optional gridspec to plot array as part of a subfigure Default: None

Returns

fig: matplotlib.figure.Figure New or original (if supplied) figure object

axes: pandas.Series(matplotlib.axes.Axes) Pandas array of axes objects

`anesthetic.plot.make_2d_axes` (*params*, ***kwargs*)
 Create a set of axes for plotting 2D marginalised posteriors.

Parameters

params: lists of parameters

Can be either: * list(str) if the x and y axes are the same * [list(str),list(str)] if the x and y axes are different Strings indicate the names of the parameters

tex: dict(str:str), optional Dictionary mapping params to tex plot labels. Default: params

upper, lower, diagonal: logical, optional Whether to create 2D marginalised plots above or below the diagonal, or to create a 1D marginalised plot on the diagonal. Default: True

fig: matplotlib.figure.Figure, optional Figure to plot on. Default: matplotlib.pyplot.figure()

subplot_spec: matplotlib.gridspec.GridSpec, optional gridspec to plot array as part of a subfigure. Default: None

Returns

fig: matplotlib.figure.Figure New or original (if supplied) figure object

axes: pandas.DataFrame(matplotlib.axes.Axes) Pandas array of axes objects

`anesthetic.plot.make_diagonal` (*ax*)
 Link x and y axes limits.

`anesthetic.plot.normalize_kwargs` (*kwargs*, *alias_mapping=None*, *drop=None*)
 Normalize kwarg inputs.

Works the same way as `cbook.normalize_kwargs`, but additionally allows to drop kwargs.

`anesthetic.plot.quantile_plot_interval` (*q*)
 Interpret quantile *q* input to quantile plot range tuple.

`anesthetic.plot.scatter_plot_2d` (*ax*, *data_x*, *data_y*, **args*, ***kwargs*)
 Plot samples from a 2d marginalised distribution.

This functions as a wrapper around `matplotlib.axes.Axes.plot`, enforcing any prior bounds. All remaining keyword arguments are passed onwards.

Parameters

ax: `matplotlib.axes.Axes` axis object to plot on

data_x, data_y: `np.array` x and y coordinates of uniformly weighted samples to plot.

xmin, xmax, ymin, ymax: `float` lower/upper prior bounds in x/y coordinates optional, default None

Returns

lines: `matplotlib.lines.Line2D` A list of line objects representing the plotted data (same as `matplotlib matplotlib.axes.Axes.plot` command)

7.5 anesthetic.samples module

Main classes for the anesthetic module.

- `MCMCSamples`
- `NestedSamples`

class `anesthetic.samples.MCMCSamples` (**args*, ***kwargs*)
 Bases: `anesthetic.weighted_pandas.WeightedDataFrame`

Storage and plotting tools for MCMC samples.

Extends the `pandas.DataFrame` by providing plotting methods and standardising sample storage.

Example plotting commands include

- `mcmc.plot_1d(['paramA', 'paramB'])`
- `mcmc.plot_2d(['paramA', 'paramB'])`
- `mcmc.plot_2d(['paramA', 'paramB'], ['paramC', 'paramD'])`

Parameters

root: `str`, **optional** root for reading chains from file. Overrides all other arguments.

data: `np.array` Coordinates of samples. shape = (nsamples, ndims).

columns: `list(str)` reference names of parameters

weights: `np.array` weights of samples.

logL: `np.array` loglikelihoods of samples.

tex: `dict` mapping from columns to tex labels for plotting

limits: dict mapping from columns to prior limits

label: str Legend label

logzero: float The threshold for $\log(0)$ values assigned to rejected sample points. Anything equal or below this value is set to $-np.inf$. default: $-1e30$

burn_in: int or float Discards the first integer number of nsamples if int or the first fraction of nsamples if float. Only works if *root* provided and if chains are GetDist compatible. default: False

Attributes

T Transpose index and columns.

at Access a single value for a row/column label pair.

axes Return a list representing the axes of the DataFrame.

blocks Internal property, property synonym for `as_blocks()`.

columns The column labels of the DataFrame.

dtypes Return the dtypes in the DataFrame.

empty Indicator whether DataFrame is empty.

ftypes Return the ftypes (indication of sparse/dense and dtype) in DataFrame.

iat Access a single value for a row/column pair by integer position.

iloc Purely integer-location based indexing for selection by position.

index The index (row labels) of the DataFrame.

is_copy Return the copy.

ix A primarily label-location based indexer, with integer position fallback.

loc Access a group of rows and columns by label(s) or a boolean array.

ndim Return an int representing the number of axes / array dimensions.

shape Return a tuple representing the dimensionality of the DataFrame.

size Return an int representing the number of elements in this object.

style Property returning a Styler object containing methods for building a styled HTML representation fo the DataFrame.

values Return a Numpy representation of the DataFrame.

weights Sample weights.

Methods

<code>abs()</code>	Return a Series/DataFrame with absolute numeric value of each element.
<code>add(other[, axis, level, fill_value])</code>	Get Addition of dataframe and other, element-wise (binary operator <i>add</i>).
<code>add_prefix(prefix)</code>	Prefix labels with string <i>prefix</i> .
<code>add_suffix(suffix)</code>	Suffix labels with string <i>suffix</i> .

Continued on next page

Table 9 – continued from previous page

<code>agg(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>aggregate(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>align(other[, join, axis, level, copy, ...])</code>	Align two objects on their axes with the specified join method for each axis Index.
<code>all([axis, bool_only, skipna, level])</code>	Return whether all elements are True, potentially over an axis.
<code>any([axis, bool_only, skipna, level])</code>	Return whether any element is True, potentially over an axis.
<code>append(other[, ignore_index, ...])</code>	Append rows of <i>other</i> to the end of caller, returning a new object.
<code>apply(func[, axis, broadcast, raw, reduce, ...])</code>	Apply a function along an axis of the DataFrame.
<code>applymap(func)</code>	Apply a function to a Dataframe elementwise.
<code>as_blocks([copy])</code>	Convert the frame to a dict of dtype -> Constructor Types that each has a homogeneous dtype.
<code>as_matrix([columns])</code>	Convert the frame to its Numpy-array representation.
<code>asfreq(freq[, method, how, normalize, ...])</code>	Convert TimeSeries to specified frequency.
<code>asof(when[, subset])</code>	Return the last row(s) without any NaNs before <i>where</i> .
<code>assign(**kwargs)</code>	Assign new columns to a DataFrame.
<code>astype(dtype[, copy, errors])</code>	Cast a pandas object to a specified dtype dtype.
<code>at_time(time[, asof, axis])</code>	Select values at particular time of day (e.g.
<code>between_time(start_time, end_time[, ...])</code>	Select values between particular times of the day (e.g., 9:00-9:30 AM).
<code>bfill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='bfill'</code> .
<code>bool()</code>	Return the bool of a single element PandasObject.
<code>boxplot([column, by, ax, fontsize, rot, ...])</code>	Make a box plot from DataFrame columns.
<code>clip([lower, upper, axis, inplace])</code>	Trim values at input threshold(s).
<code>clip_lower(threshold[, axis, inplace])</code>	Trim values below a given threshold.
<code>clip_upper(threshold[, axis, inplace])</code>	Trim values above a given threshold.
<code>combine(other, func[, fill_value, overwrite])</code>	Perform column-wise combine with another DataFrame.
<code>combine_first(other)</code>	Update null elements with value in the same location in <i>other</i> .
<code>compound([axis, skipna, level])</code>	Return the compound percentage of the values for the requested axis.
<code>compress([nsamples])</code>	Reduce the number of samples by discarding low-weights.
<code>copy([deep])</code>	Make a copy of this object's indices and data.
<code>corr([method, min_periods])</code>	Compute pairwise correlation of columns, excluding NA/null values.
<code>corrwith(other[, axis, drop, method])</code>	Compute pairwise correlation between rows or columns of DataFrame with rows or columns of Series or DataFrame.
<code>count([axis, level, numeric_only])</code>	Count non-NA cells for each column or row.
<code>cov()</code>	Weighted covariance of the sampled distribution.
<code>cummax([axis, skipna])</code>	Return cumulative maximum over a DataFrame or Series axis.

Continued on next page

Table 9 – continued from previous page

<code>cummin([axis, skipna])</code>	Return cumulative minimum over a DataFrame or Series axis.
<code>cumprod([axis, skipna])</code>	Return cumulative product over a DataFrame or Series axis.
<code>cumsum([axis, skipna])</code>	Return cumulative sum over a DataFrame or Series axis.
<code>describe([percentiles, include, exclude])</code>	Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.
<code>diff([periods, axis])</code>	First discrete difference of element.
<code>div(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>divide(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>dot(other)</code>	Compute the matrix multiplication between the DataFrame and other.
<code>drop([labels, axis, index, columns, level, ...])</code>	Drop specified labels from rows or columns.
<code>drop_duplicates([subset, keep, inplace])</code>	Return DataFrame with duplicate rows removed, optionally only considering certain columns.
<code>droplevel(level[, axis])</code>	Return DataFrame with requested index / column level(s) removed.
<code>dropna([axis, how, thresh, subset, inplace])</code>	Remove missing values.
<code>duplicated([subset, keep])</code>	Return boolean Series denoting duplicate rows, optionally only considering certain columns.
<code>eq(other[, axis, level])</code>	Get Equal to of dataframe and other, element-wise (binary operator <i>eq</i>).
<code>equals(other)</code>	Test whether two objects contain the same elements.
<code>eval(expr[, inplace])</code>	Evaluate a string describing operations on DataFrame columns.
<code>ewm([com, span, halflife, alpha, ...])</code>	Provide exponential weighted functions.
<code>expanding([min_periods, center, axis])</code>	Provide expanding transformations.
<code>explode(column, Tuple)</code>	Transform each element of a list-like to a row, replicating the index values.
<code>ffill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='ffill'</code> .
<code>fillna([value, method, axis, inplace, ...])</code>	Fill NA/NaN values using the specified method.
<code>filter([items, like, regex, axis])</code>	Subset rows or columns of dataframe according to labels in the specified index.
<code>first(offset)</code>	Convenience method for subsetting initial periods of time series data based on a date offset.
<code>first_valid_index()</code>	Return index for first non-NA/null value.
<code>floordiv(other[, axis, level, fill_value])</code>	Get Integer division of dataframe and other, element-wise (binary operator <i>floordiv</i>).
<code>from_dict(data[, orient, dtype, columns])</code>	Construct DataFrame from dict of array-like or dicts.
<code>from_items(items[, columns, orient])</code>	Construct a DataFrame from a list of tuples.
<code>from_records(data[, index, exclude, ...])</code>	Convert structured or record ndarray to DataFrame.
<code>ge(other[, axis, level])</code>	Get Greater than or equal to of dataframe and other, element-wise (binary operator <i>ge</i>).
<code>get(key[, default])</code>	Get item from object for given key (ex: DataFrame column).
<code>get_dtype_counts()</code>	Return counts of unique dtypes in this object.

Continued on next page

Table 9 – continued from previous page

<code>get_ftype_counts()</code>	Return counts of unique ftypes in this object.
<code>get_value(index, col[, takeable])</code>	Quickly retrieve single value at passed column and index.
<code>get_values()</code>	Return an ndarray after converting sparse values to dense.
<code>groupby([by, axis, level, as_index, sort, ...])</code>	Group DataFrame or Series using a mapper or by a Series of columns.
<code>gt(other[, axis, level])</code>	Get Greater than of dataframe and other, element-wise (binary operator <i>gt</i>).
<code>head([n])</code>	Return the first <i>n</i> rows.
<code>hist(*args, **kwargs)</code>	Weighted histogram of the sampled distribution.
<code>idxmax([axis, skipna])</code>	Return index of first occurrence of maximum over requested axis.
<code>idxmin([axis, skipna])</code>	Return index of first occurrence of minimum over requested axis.
<code>infer_objects()</code>	Attempt to infer better dtypes for object columns.
<code>info([verbose, buf, max_cols, memory_usage, ...])</code>	Print a concise summary of a DataFrame.
<code>insert(loc, column, value[, allow_duplicates])</code>	Insert column into DataFrame at specified location.
<code>interpolate([method, axis, limit, inplace, ...])</code>	Interpolate values according to different methods.
<code>isin(values)</code>	Whether each element in the DataFrame is contained in values.
<code>isna()</code>	Detect missing values.
<code>isnull()</code>	Detect missing values.
<code>items()</code>	Iterator over (column name, Series) pairs.
<code>iteritems()</code>	Iterator over (column name, Series) pairs.
<code>iterrows()</code>	Iterate over DataFrame rows as (index, Series) pairs.
<code>itertuples([index, name])</code>	Iterate over DataFrame rows as namedtuples.
<code>join(other[, on, how, lsuffix, rsuffix, sort])</code>	Join columns of another DataFrame.
<code>keys()</code>	Get the ‘info axis’ (see Indexing for more)
<code>kurt([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher’s definition of kurtosis (kurtosis of normal == 0.0).
<code>kurtosis([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher’s definition of kurtosis (kurtosis of normal == 0.0).
<code>last(offset)</code>	Convenience method for subsetting final periods of time series data based on a date offset.
<code>last_valid_index()</code>	Return index for last non-NA/null value.
<code>le(other[, axis, level])</code>	Get Less than or equal to of dataframe and other, element-wise (binary operator <i>le</i>).
<code>lookup(row_labels, col_labels)</code>	Label-based “fancy indexing” function for DataFrame.
<code>lt(other[, axis, level])</code>	Get Less than of dataframe and other, element-wise (binary operator <i>lt</i>).
<code>mad([axis, skipna, level])</code>	Return the mean absolute deviation of the values for the requested axis.
<code>mask(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is True.
<code>max([axis, skipna, level, numeric_only])</code>	Return the maximum of the values for the requested axis.
<code>mean()</code>	Weighted mean of the sampled distribution.

Continued on next page

Table 9 – continued from previous page

<code>median()</code>	Weighted median of the sampled distribution.
<code>melt([id_vars, value_vars, var_name, ...])</code>	Unpivot a DataFrame from wide format to long format, optionally leaving identifier variables set.
<code>memory_usage([index, deep])</code>	Return the memory usage of each column in bytes.
<code>merge(right[, how, on, left_on, right_on, ...])</code>	Merge DataFrame or named Series objects with a database-style join.
<code>min([axis, skipna, level, numeric_only])</code>	Return the minimum of the values for the requested axis.
<code>mod(other[, axis, level, fill_value])</code>	Get Modulo of dataframe and other, element-wise (binary operator <i>mod</i>).
<code>mode([axis, numeric_only, dropna])</code>	Get the mode(s) of each element along the selected axis.
<code>mul(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>mul</i>).
<code>multiply(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>mul</i>).
<code>ne(other[, axis, level])</code>	Get Not equal to of dataframe and other, element-wise (binary operator <i>ne</i>).
<code>neff()</code>	Effective number of samples.
<code>nlargest(n, columns[, keep])</code>	Return the first <i>n</i> rows ordered by <i>columns</i> in descending order.
<code>notna()</code>	Detect existing (non-missing) values.
<code>notnull()</code>	Detect existing (non-missing) values.
<code>nsmallest(n, columns[, keep])</code>	Return the first <i>n</i> rows ordered by <i>columns</i> in ascending order.
<code>nunique([axis, dropna])</code>	Count distinct observations over requested axis.
<code>pct_change([periods, fill_method, limit, freq])</code>	Percentage change between the current and a prior element.
<code>pipe(func, *args, **kwargs)</code>	Apply <code>func(self, *args, **kwargs)</code> .
<code>pivot([index, columns, values])</code>	Return reshaped DataFrame organized by given index / column values.
<code>pivot_table([values, index, columns, ...])</code>	Create a spreadsheet-style pivot table as a DataFrame.
<code>plot(ax, paramname_x[, paramname_y])</code>	Interface for 2D and 1D plotting routines.
<code>plot_1d(axes, *args, **kwargs)</code>	Create an array of 1D plots.
<code>plot_2d(axes, *args, **kwargs)</code>	Create an array of 2D plots.
<code>pop(item)</code>	Return item and drop from frame.
<code>pow(other[, axis, level, fill_value])</code>	Get Exponential power of dataframe and other, element-wise (binary operator <i>pow</i>).
<code>prod([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>product([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>quantile([q])</code>	Weighted quantile of the sampled distribution.
<code>query(expr[, inplace])</code>	Query the columns of a DataFrame with a boolean expression.
<code>radd(other[, axis, level, fill_value])</code>	Get Addition of dataframe and other, element-wise (binary operator <i>radd</i>).
<code>rank([axis, method, numeric_only, ...])</code>	Compute numerical data ranks (1 through <i>n</i>) along axis.

Continued on next page

Table 9 – continued from previous page

<code>rdiv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>rtruediv</i>).
<code>reindex([labels, index, columns, axis, ...])</code>	Conform DataFrame to new index with optional filling logic, placing NA/NaN in locations having no value in the previous index.
<code>reindex_like(other[, method, copy, limit, ...])</code>	Return an object with matching indices as other object.
<code>rename([mapper, index, columns, axis, copy, ...])</code>	Alter axes labels.
<code>rename_axis([mapper, index, columns, axis, ...])</code>	Set the name of the axis for the index or columns.
<code>reorder_levels(order[, axis])</code>	Rearrange index levels using input order.
<code>replace([to_replace, value, inplace, limit, ...])</code>	Replace values given in <i>to_replace</i> with <i>value</i> .
<code>resample(rule[, how, axis, fill_method, ...])</code>	Resample time-series data.
<code>reset_index([level, drop, inplace, ...])</code>	Reset the index, or a level of it.
<code>rfloordiv(other[, axis, level, fill_value])</code>	Get Integer division of dataframe and other, element-wise (binary operator <i>rfloordiv</i>).
<code>rmod(other[, axis, level, fill_value])</code>	Get Modulo of dataframe and other, element-wise (binary operator <i>rmod</i>).
<code>rmul(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>rmul</i>).
<code>rolling(window[, min_periods, center, ...])</code>	Provide rolling window calculations.
<code>round([decimals])</code>	Round a DataFrame to a variable number of decimal places.
<code>rpow(other[, axis, level, fill_value])</code>	Get Exponential power of dataframe and other, element-wise (binary operator <i>rpow</i>).
<code>rsub(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>rsub</i>).
<code>rtruediv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>rtruediv</i>).
<code>sample([n, frac, replace, weights, ...])</code>	Return a random sample of items from an axis of object.
<code>select_dtypes([include, exclude])</code>	Return a subset of the DataFrame's columns based on the column dtypes.
<code>sem([axis, skipna, level, ddof, numeric_only])</code>	Return unbiased standard error of the mean over requested axis.
<code>set_axis(labels[, axis, inplace])</code>	Assign desired index to given axis.
<code>set_index(keys[, drop, append, inplace, ...])</code>	Set the DataFrame index using existing columns.
<code>set_value(index, col, value[, takeable])</code>	Put single value at passed column and index.
<code>shift([periods, freq, axis, fill_value])</code>	Shift index by desired number of periods with an optional time <i>freq</i> .
<code>skew([axis, skipna, level, numeric_only])</code>	Return unbiased skew over requested axis Normalized by N-1.
<code>slice_shift([periods, axis])</code>	Equivalent to <i>shift</i> without copying data.
<code>sort_index([axis, level, ascending, ...])</code>	Sort object by labels (along an axis).
<code>sort_values(by[, axis, ascending, inplace, ...])</code>	Sort by the values along either axis.
<code>sparse</code>	alias of <code>pandas.core.arrays.sparse.SparseFrameAccessor</code>
<code>squeeze([axis])</code>	Squeeze 1 dimensional axis objects into scalars.
<code>stack([level, dropna])</code>	Stack the prescribed level(s) from columns to index.
<code>std()</code>	Weighted standard deviation of the sampled distribution.

Continued on next page

Table 9 – continued from previous page

<code>sub(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>sub</i>).
<code>subtract(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>sub</i>).
<code>sum([axis, skipna, level, numeric_only, ...])</code>	Return the sum of the values for the requested axis.
<code>swapaxes(axis1, axis2[, copy])</code>	Interchange axes and swap values axes appropriately.
<code>swaplevel([i, j, axis])</code>	Swap levels i and j in a MultiIndex on a particular axis.
<code>tail([n])</code>	Return the last <i>n</i> rows.
<code>take(indices[, axis, is_copy])</code>	Return the elements in the given <i>positional</i> indices along an axis.
<code>to_clipboard([excel, sep])</code>	Copy object to the system clipboard.
<code>to_csv([path_or_buf, sep, na_rep, ...])</code>	Write object to a comma-separated values (csv) file.
<code>to_dense()</code>	Return dense representation of Series/DataFrame (as opposed to sparse).
<code>to_dict([orient, into])</code>	Convert the DataFrame to a dictionary.
<code>to_excel(excel_writer[, sheet_name, na_rep, ...])</code>	Write object to an Excel sheet.
<code>to_feather(fname)</code>	Write out the binary feather-format for DataFrames.
<code>to_gbq(destination_table[, project_id, ...])</code>	Write a DataFrame to a Google BigQuery table.
<code>to_hdf(path_or_buf, key, **kwargs)</code>	Write the contained data to an HDF5 file using HDF-Store.
<code>to_html([buf, columns, col_space, header, ...])</code>	Render a DataFrame as an HTML table.
<code>to_json([path_or_buf, orient, date_format, ...])</code>	Convert the object to a JSON string.
<code>to_latex([buf, columns, col_space, header, ...])</code>	Render an object to a LaTeX tabular environment table.
<code>to_msgpack([path_or_buf, encoding])</code>	Serialize object to input file path using msgpack format.
<code>to_numpy([dtype, copy])</code>	Convert the DataFrame to a NumPy array.
<code>to_parquet(fname[, engine, compression, ...])</code>	Write a DataFrame to the binary parquet format.
<code>to_period([freq, axis, copy])</code>	Convert DataFrame from DatetimeIndex to PeriodIndex with desired frequency (inferred from index if not passed).
<code>to_pickle(path[, compression, protocol])</code>	Pickle (serialize) object to file.
<code>to_records([index, convert_datetime64, ...])</code>	Convert DataFrame to a NumPy record array.
<code>to_sparse([fill_value, kind])</code>	Convert to SparseDataFrame.
<code>to_sql(name, con[, schema, if_exists, ...])</code>	Write records stored in a DataFrame to a SQL database.
<code>to_stata(fname[, convert_dates, ...])</code>	Export DataFrame object to Stata dta format.
<code>to_string([buf, columns, col_space, header, ...])</code>	Render a DataFrame to a console-friendly tabular output.
<code>to_timestamp([freq, how, axis, copy])</code>	Cast to DatetimeIndex of timestamps, at <i>beginning</i> of period.
<code>to_xarray()</code>	Return an xarray object from the pandas object.
<code>transform(func[, axis])</code>	Call <i>func</i> on self producing a DataFrame with transformed values and that has the same axis length as self.
<code>transpose(*args, **kwargs)</code>	Transpose index and columns.
<code>truediv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).

Continued on next page

Table 9 – continued from previous page

<code>truncate([before, after, axis, copy])</code>	Truncate a Series or DataFrame before and after some index value.
<code>tshift([periods, freq, axis])</code>	Shift the time index, using the index’s frequency if available.
<code>tz_convert(tz[, axis, level, copy])</code>	Convert tz-aware axis to target time zone.
<code>tz_localize(tz[, axis, level, copy, ...])</code>	Localize tz-naive index of a Series or DataFrame to target time zone.
<code>unstack([level, fill_value])</code>	Pivot a level of the (necessarily hierarchical) index labels, returning a DataFrame having a new level of column labels whose inner-most level consists of the pivoted index labels.
<code>update(other[, join, overwrite, ...])</code>	Modify in place using non-NA values from another DataFrame.
<code>var()</code>	Weighted variance of the sampled distribution.
<code>where(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is False.
<code>xs(key[, axis, level, drop_level])</code>	Return cross-section from the Series/DataFrame.

plot (*ax*, *paramname_x*, *paramname_y=None*, **args*, ***kwargs*)

Interface for 2D and 1D plotting routines.

Produces a single 1D or 2D plot on an axis.

Parameters

ax: `matplotlib.axes.Axes` Axes to plot on

paramname_x: `str` Choice of parameter to plot on x-coordinate from `self.columns`.

paramname_y: `str` Choice of parameter to plot on y-coordinate from `self.columns`. optional, if not provided, or the same as `paramname_x`, then 1D plot produced.

plot_type: `str` Must be in `{‘kde’, ‘scatter’, ‘hist’, ‘fastkde’}` for 2D plots and in `{‘kde’, ‘hist’, ‘fastkde’, ‘astropyhist’}` for 1D plots. optional, (Default: ‘kde’)

ncompress: `int` Number of samples to use in plotting routines. optional, Default dynamically chosen

q: `str, float, (float, float)` Plot the *q* inner posterior quantiles in 1d ‘kde’ plots. To plot the full range, set *q=0* or *q=1*. * if `str`: any of `{‘1sigma’, ‘2sigma’, ‘3sigma’, ‘4sigma’, ‘5sigma’}`

Plot within mean +/- #sigma of posterior.

- **if float:** Plot within the symmetric confidence interval $(1-q, q)$ or $(q, 1-q)$.
- **if tuple:** Plot within the (possibly asymmetric) confidence interval *q*.

optional, (Default: ‘5sigma’)

Returns

fig: `matplotlib.figure.Figure` New or original (if supplied) figure object

axes: `pandas.DataFrame` or `pandas.Series` of `matplotlib.axes.Axes` Pandas array of axes objects

plot_1d (*axes*, **args*, ***kwargs*)

Create an array of 1D plots.

Parameters

axes: plotting axes

Can be:

- list(str) or str
- pandas.Series(matplotlib.axes.Axes)

If a pandas.Series is provided as an existing set of axes, then this is used for creating the plot. Otherwise a new set of axes are created using the list or lists of strings.

Returns

fig: matplotlib.figure.Figure New or original (if supplied) figure object

axes: pandas.Series of matplotlib.axes.Axes Pandas array of axes objects

plot_2d (*axes*, *args, **kwargs)

Create an array of 2D plots.

To avoid interfering with y-axis sharing, one-dimensional plots are created on a separate axis, which is monkey-patched onto the argument ax as the attribute ax.twin.

Parameters

axes: plotting axes

Can be:

- list(str) if the x and y axes are the same
- [list(str),list(str)] if the x and y axes are different
- pandas.DataFrame(matplotlib.axes.Axes)

If a pandas.DataFrame is provided as an existing set of axes, then this is used for creating the plot. Otherwise a new set of axes are created using the list or lists of strings.

types: dict, optional What type (or types) of plots to produce. Takes the keys ‘diagonal’ for the 1D plots and ‘lower’ and ‘upper’ for the 2D plots. The options for ‘diagonal’ are:

- ‘kde’
- ‘hist’
- ‘astropyhist’

The options for ‘lower’ and ‘upper’ are:

- ‘kde’
- ‘scatter’
- ‘hist’
- ‘fastkde’

Default: { ‘diagonal’: ‘kde’, ‘lower’: ‘kde’, ‘upper’: ‘scatter’ }

diagonal_kwargs, lower_kwargs, upper_kwargs: dict, optional kwargs for the diagonal (1D)/lower or upper (2D) plots. This is useful when there is a conflict of kwargs for different types of plots. Note that any kwargs directly passed to plot_2d will overwrite any kwarg with the same key passed to <sub><sub>_kwargs. Default: { }

Returns

fig: `matplotlib.figure.Figure` New or original (if supplied) figure object

axes: `pandas.DataFrame` of `matplotlib.axes.Axes` Pandas array of axes objects

class `anesthetic.samples.NestedSamples` (*args, **kwargs)

Bases: `anesthetic.samples.MCMCSamples`

Storage and plotting tools for Nested Sampling samples.

We extend the MCMCSamples class with the additional methods:

- `self.ns_output()`
- `self.live_points(logL)`
- `self.posterior_points(beta)`

Parameters

root: `str`, **optional** root for reading chains from file. Overrides all other arguments.

data: `np.array` Coordinates of samples. shape = (nsamples, ndims).

columns: `list(str)` reference names of parameters

logL: `np.array` loglikelihoods of samples.

logL_birth: `np.array` or `int` birth loglikelihoods, or number of live points.

tex: `dict` optional mapping from column names to tex labels for plotting

limits: `dict` mapping from columns to prior limits. Defaults defined by `.ranges` file (if it exists) otherwise defined by minimum and maximum of the nested sampling data

label: `str` Legend label default: basename of root

beta: `float` thermodynamic temperature default: 1.

logzero: `float` The threshold for $\log(0)$ values assigned to rejected sample points. Anything equal or below this value is set to `-np.inf`. default: `-1e30`

Attributes

T Transpose index and columns.

at Access a single value for a row/column label pair.

axes Return a list representing the axes of the DataFrame.

beta Thermodynamic inverse temperature.

blocks Internal property, property synonym for `as_blocks()`.

columns The column labels of the DataFrame.

dtypes Return the dtypes in the DataFrame.

empty Indicator whether DataFrame is empty.

ftypes Return the ftypes (indication of sparse/dense and dtype) in DataFrame.

iat Access a single value for a row/column pair by integer position.

iloc Purely integer-location based indexing for selection by position.

index The index (row labels) of the DataFrame.

is_copy Return the copy.

ix A primarily label-location based indexer, with integer position fallback.

- loc** Access a group of rows and columns by label(s) or a boolean array.
- ndim** Return an int representing the number of axes / array dimensions.
- shape** Return a tuple representing the dimensionality of the DataFrame.
- size** Return an int representing the number of elements in this object.
- style** Property returning a Styler object containing methods for building a styled HTML representation for the DataFrame.
- values** Return a Numpy representation of the DataFrame.
- weights** Sample weights.

Methods

<code>D([nsamples])</code>	Kullback-Leibler divergence.
<code>abs()</code>	Return a Series/DataFrame with absolute numeric value of each element.
<code>add(other[, axis, level, fill_value])</code>	Get Addition of dataframe and other, element-wise (binary operator <i>add</i>).
<code>add_prefix(prefix)</code>	Prefix labels with string <i>prefix</i> .
<code>add_suffix(suffix)</code>	Suffix labels with string <i>suffix</i> .
<code>agg(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>aggregate(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>align(other[, join, axis, level, copy, ...])</code>	Align two objects on their axes with the specified join method for each axis Index.
<code>all([axis, bool_only, skipna, level])</code>	Return whether all elements are True, potentially over an axis.
<code>any([axis, bool_only, skipna, level])</code>	Return whether any element is True, potentially over an axis.
<code>append(other[, ignore_index, ...])</code>	Append rows of <i>other</i> to the end of caller, returning a new object.
<code>apply(func[, axis, broadcast, raw, reduce, ...])</code>	Apply a function along an axis of the DataFrame.
<code>applymap(func)</code>	Apply a function to a Dataframe elementwise.
<code>as_blocks([copy])</code>	Convert the frame to a dict of dtype -> Constructor Types that each has a homogeneous dtype.
<code>as_matrix([columns])</code>	Convert the frame to its Numpy-array representation.
<code>asfreq(freq[, method, how, normalize, ...])</code>	Convert TimeSeries to specified frequency.
<code>asof(when[, subset])</code>	Return the last row(s) without any NaNs before <i>when</i> .
<code>assign(**kwargs)</code>	Assign new columns to a DataFrame.
<code>astype(dtype[, copy, errors])</code>	Cast a pandas object to a specified dtype <i>dtype</i> .
<code>at_time(time[, asof, axis])</code>	Select values at particular time of day (e.g.
<code>between_time(start_time, end_time[, ...])</code>	Select values between particular times of the day (e.g., 9:00-9:30 AM).
<code>bfill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='bfill'</code> .
<code>bool()</code>	Return the bool of a single element PandasObject.
<code>boxplot([column, by, ax, fontsize, rot, ...])</code>	Make a box plot from DataFrame columns.
<code>clip([lower, upper, axis, inplace])</code>	Trim values at input threshold(s).

Continued on next page

Table 10 – continued from previous page

<code>clip_lower(threshold[, axis, inplace])</code>	Trim values below a given threshold.
<code>clip_upper(threshold[, axis, inplace])</code>	Trim values above a given threshold.
<code>combine(other, func[, fill_value, overwrite])</code>	Perform column-wise combine with another DataFrame.
<code>combine_first(other)</code>	Update null elements with value in the same location in <i>other</i> .
<code>compound([axis, skipna, level])</code>	Return the compound percentage of the values for the requested axis.
<code>compress([nsamples])</code>	Reduce the number of samples by discarding low-weights.
<code>copy([deep])</code>	Make a copy of this object's indices and data.
<code>corr([method, min_periods])</code>	Compute pairwise correlation of columns, excluding NA/null values.
<code>corrwith(other[, axis, drop, method])</code>	Compute pairwise correlation between rows or columns of DataFrame with rows or columns of Series or DataFrame.
<code>count([axis, level, numeric_only])</code>	Count non-NA cells for each column or row.
<code>cov()</code>	Weighted covariance of the sampled distribution.
<code>cummax([axis, skipna])</code>	Return cumulative maximum over a DataFrame or Series axis.
<code>cummin([axis, skipna])</code>	Return cumulative minimum over a DataFrame or Series axis.
<code>cumprod([axis, skipna])</code>	Return cumulative product over a DataFrame or Series axis.
<code>cumsum([axis, skipna])</code>	Return cumulative sum over a DataFrame or Series axis.
<code>d([nsamples])</code>	Bayesian model dimensionality.
<code>describe([percentiles, include, exclude])</code>	Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.
<code>diff([periods, axis])</code>	First discrete difference of element.
<code>div(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>divide(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>dlogX([nsamples])</code>	Compute volume of shell of loglikelihood.
<code>dot(other)</code>	Compute the matrix multiplication between the DataFrame and other.
<code>drop([labels, axis, index, columns, level, ...])</code>	Drop specified labels from rows or columns.
<code>drop_duplicates([subset, keep, inplace])</code>	Return DataFrame with duplicate rows removed, optionally only considering certain columns.
<code>droplevel(level[, axis])</code>	Return DataFrame with requested index / column level(s) removed.
<code>dropna([axis, how, thresh, subset, inplace])</code>	Remove missing values.
<code>duplicated([subset, keep])</code>	Return boolean Series denoting duplicate rows, optionally only considering certain columns.
<code>eq(other[, axis, level])</code>	Get Equal to of dataframe and other, element-wise (binary operator <i>eq</i>).
<code>equals(other)</code>	Test whether two objects contain the same elements.
<code>eval(expr[, inplace])</code>	Evaluate a string describing operations on DataFrame columns.

Continued on next page

Table 10 – continued from previous page

<code>ewm([com, span, halflife, alpha, ...])</code>	Provide exponential weighted functions.
<code>expanding([min_periods, center, axis])</code>	Provide expanding transformations.
<code>explode(column, Tuple)</code>	Transform each element of a list-like to a row, replicating the index values.
<code>ffill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='ffill'</code> .
<code>fillna([value, method, axis, inplace, ...])</code>	Fill NA/NaN values using the specified method.
<code>filter([items, like, regex, axis])</code>	Subset rows or columns of dataframe according to labels in the specified index.
<code>first(offset)</code>	Convenience method for subsetting initial periods of time series data based on a date offset.
<code>first_valid_index()</code>	Return index for first non-NA/null value.
<code>floordiv(other[, axis, level, fill_value])</code>	Get Integer division of dataframe and other, element-wise (binary operator <i>floordiv</i>).
<code>from_dict(data[, orient, dtype, columns])</code>	Construct DataFrame from dict of array-like or dicts.
<code>from_items(items[, columns, orient])</code>	Construct a DataFrame from a list of tuples.
<code>from_records(data[, index, exclude, ...])</code>	Convert structured or record ndarray to DataFrame.
<code>ge(other[, axis, level])</code>	Get Greater than or equal to of dataframe and other, element-wise (binary operator <i>ge</i>).
<code>get(key[, default])</code>	Get item from object for given key (ex: DataFrame column).
<code>get_dtype_counts()</code>	Return counts of unique dtypes in this object.
<code>get_ftype_counts()</code>	Return counts of unique ftypes in this object.
<code>get_value(index, col[, takeable])</code>	Quickly retrieve single value at passed column and index.
<code>get_values()</code>	Return an ndarray after converting sparse values to dense.
<code>groupby([by, axis, level, as_index, sort, ...])</code>	Group DataFrame or Series using a mapper or by a Series of columns.
<code>gt(other[, axis, level])</code>	Get Greater than of dataframe and other, element-wise (binary operator <i>gt</i>).
<code>gui([params])</code>	Construct a graphical user interface for viewing samples.
<code>head([n])</code>	Return the first <i>n</i> rows.
<code>hist(*args, **kwargs)</code>	Weighted histogram of the sampled distribution.
<code>idxmax([axis, skipna])</code>	Return index of first occurrence of maximum over requested axis.
<code>idxmin([axis, skipna])</code>	Return index of first occurrence of minimum over requested axis.
<code>infer_objects()</code>	Attempt to infer better dtypes for object columns.
<code>info([verbose, buf, max_cols, memory_usage, ...])</code>	Print a concise summary of a DataFrame.
<code>insert(loc, column, value[, allow_duplicates])</code>	Insert column into DataFrame at specified location.
<code>interpolate([method, axis, limit, inplace, ...])</code>	Interpolate values according to different methods.
<code>isin(values)</code>	Whether each element in the DataFrame is contained in values.
<code>isna()</code>	Detect missing values.
<code>isnull()</code>	Detect missing values.
<code>items()</code>	Iterator over (column name, Series) pairs.
<code>iteritems()</code>	Iterator over (column name, Series) pairs.
<code>iterrows()</code>	Iterate over DataFrame rows as (index, Series) pairs.

Continued on next page

Table 10 – continued from previous page

<code>itertuples([index, name])</code>	Iterate over DataFrame rows as namedtuples.
<code>join(other[, on, how, lsuffix, rsuffix, sort])</code>	Join columns of another DataFrame.
<code>keys()</code>	Get the ‘info axis’ (see Indexing for more)
<code>kurt([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher’s definition of kurtosis (kurtosis of normal == 0.0).
<code>kurtosis([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher’s definition of kurtosis (kurtosis of normal == 0.0).
<code>last(offset)</code>	Convenience method for subsetting final periods of time series data based on a date offset.
<code>last_valid_index()</code>	Return index for last non-NA/null value.
<code>le(other[, axis, level])</code>	Get Less than or equal to of dataframe and other, element-wise (binary operator <i>le</i>).
<code>live_points([logL])</code>	Get the live points within logL.
<code>logZ([nsamples])</code>	Log-Evidence.
<code>lookup(row_labels, col_labels)</code>	Label-based “fancy indexing” function for DataFrame.
<code>lt(other[, axis, level])</code>	Get Less than of dataframe and other, element-wise (binary operator <i>lt</i>).
<code>mad([axis, skipna, level])</code>	Return the mean absolute deviation of the values for the requested axis.
<code>mask(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is True.
<code>max([axis, skipna, level, numeric_only])</code>	Return the maximum of the values for the requested axis.
<code>mean()</code>	Weighted mean of the sampled distribution.
<code>median()</code>	Weighted median of the sampled distribution.
<code>melt([id_vars, value_vars, var_name, ...])</code>	Unpivot a DataFrame from wide format to long format, optionally leaving identifier variables set.
<code>memory_usage([index, deep])</code>	Return the memory usage of each column in bytes.
<code>merge(right[, how, on, left_on, right_on, ...])</code>	Merge DataFrame or named Series objects with a database-style join.
<code>min([axis, skipna, level, numeric_only])</code>	Return the minimum of the values for the requested axis.
<code>mod(other[, axis, level, fill_value])</code>	Get Modulo of dataframe and other, element-wise (binary operator <i>mod</i>).
<code>mode([axis, numeric_only, dropna])</code>	Get the mode(s) of each element along the selected axis.
<code>mul(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>mul</i>).
<code>multiply(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>mul</i>).
<code>ne(other[, axis, level])</code>	Get Not equal to of dataframe and other, element-wise (binary operator <i>ne</i>).
<code>neff()</code>	Effective number of samples.
<code>nlargest(n, columns[, keep])</code>	Return the first <i>n</i> rows ordered by <i>columns</i> in descending order.
<code>notna()</code>	Detect existing (non-missing) values.
<code>notnull()</code>	Detect existing (non-missing) values.
<code>ns_output([nsamples])</code>	Compute Bayesian global quantities.

Continued on next page

Table 10 – continued from previous page

<code>nsmallest(n, columns[, keep])</code>	Return the first n rows ordered by <i>columns</i> in ascending order.
<code>nunique([axis, dropna])</code>	Count distinct observations over requested axis.
<code>pct_change([periods, fill_method, limit, freq])</code>	Percentage change between the current and a prior element.
<code>pipe(func, *args, **kwargs)</code>	Apply <code>func(self, *args, **kwargs)</code> .
<code>pivot([index, columns, values])</code>	Return reshaped DataFrame organized by given index / column values.
<code>pivot_table([values, index, columns, ...])</code>	Create a spreadsheet-style pivot table as a DataFrame.
<code>plot(ax, paramname_x[, paramname_y])</code>	Interface for 2D and 1D plotting routines.
<code>plot_1d(axes, *args, **kwargs)</code>	Create an array of 1D plots.
<code>plot_2d(axes, *args, **kwargs)</code>	Create an array of 2D plots.
<code>pop(item)</code>	Return item and drop from frame.
<code>posterior_points([beta])</code>	Get equally weighted posterior points at temperature beta.
<code>pow(other[, axis, level, fill_value])</code>	Get Exponential power of dataframe and other, element-wise (binary operator <i>pow</i>).
<code>prod([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>product([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>quantile([q])</code>	Weighted quantile of the sampled distribution.
<code>query(expr[, inplace])</code>	Query the columns of a DataFrame with a boolean expression.
<code>radd(other[, axis, level, fill_value])</code>	Get Addition of dataframe and other, element-wise (binary operator <i>radd</i>).
<code>rank([axis, method, numeric_only, ...])</code>	Compute numerical data ranks (1 through n) along axis.
<code>rdiv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>rtruediv</i>).
<code>reindex([labels, index, columns, axis, ...])</code>	Conform DataFrame to new index with optional filling logic, placing NA/NaN in locations having no value in the previous index.
<code>reindex_like(other[, method, copy, limit, ...])</code>	Return an object with matching indices as other object.
<code>rename([mapper, index, columns, axis, copy, ...])</code>	Alter axes labels.
<code>rename_axis([mapper, index, columns, axis, ...])</code>	Set the name of the axis for the index or columns.
<code>reorder_levels(order[, axis])</code>	Rearrange index levels using input order.
<code>replace([to_replace, value, inplace, limit, ...])</code>	Replace values given in <i>to_replace</i> with <i>value</i> .
<code>resample(rule[, how, axis, fill_method, ...])</code>	Resample time-series data.
<code>reset_index([level, drop, inplace, ...])</code>	Reset the index, or a level of it.
<code>rfloordiv(other[, axis, level, fill_value])</code>	Get Integer division of dataframe and other, element-wise (binary operator <i>rfloordiv</i>).
<code>rmod(other[, axis, level, fill_value])</code>	Get Modulo of dataframe and other, element-wise (binary operator <i>rmod</i>).
<code>rmul(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>rmul</i>).
<code>rolling(window[, min_periods, center, ...])</code>	Provide rolling window calculations.

Continued on next page

Table 10 – continued from previous page

<code>round([decimals])</code>	Round a DataFrame to a variable number of decimal places.
<code>rpow(other[, axis, level, fill_value])</code>	Get Exponential power of dataframe and other, element-wise (binary operator <i>rpow</i>).
<code>rsub(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>rsub</i>).
<code>rtruediv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>rtruediv</i>).
<code>sample([n, frac, replace, weights, ...])</code>	Return a random sample of items from an axis of object.
<code>select_dtypes([include, exclude])</code>	Return a subset of the DataFrame’s columns based on the column dtypes.
<code>sem([axis, skipna, level, ddof, numeric_only])</code>	Return unbiased standard error of the mean over requested axis.
<code>set_axis(labels[, axis, inplace])</code>	Assign desired index to given axis.
<code>set_beta(beta[, inplace])</code>	Change the inverse temperature.
<code>set_index(keys[, drop, append, inplace, ...])</code>	Set the DataFrame index using existing columns.
<code>set_value(index, col, value[, takeable])</code>	Put single value at passed column and index.
<code>shift([periods, freq, axis, fill_value])</code>	Shift index by desired number of periods with an optional time <i>freq</i> .
<code>skew([axis, skipna, level, numeric_only])</code>	Return unbiased skew over requested axis Normalized by N-1.
<code>slice_shift([periods, axis])</code>	Equivalent to <i>shift</i> without copying data.
<code>sort_index([axis, level, ascending, ...])</code>	Sort object by labels (along an axis).
<code>sort_values(by[, axis, ascending, inplace, ...])</code>	Sort by the values along either axis.
<code>sparse</code>	alias of <code>pandas.core.arrays.sparse.SparseFrameAccessor</code>
<code>squeeze([axis])</code>	Squeeze 1 dimensional axis objects into scalars.
<code>stack([level, dropna])</code>	Stack the prescribed level(s) from columns to index.
<code>std()</code>	Weighted standard deviation of the sampled distribution.
<code>sub(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>sub</i>).
<code>subtract(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>sub</i>).
<code>sum([axis, skipna, level, numeric_only, ...])</code>	Return the sum of the values for the requested axis.
<code>swapaxes(axis1, axis2[, copy])</code>	Interchange axes and swap values axes appropriately.
<code>swaplevel([i, j, axis])</code>	Swap levels i and j in a MultiIndex on a particular axis.
<code>tail([n])</code>	Return the last <i>n</i> rows.
<code>take(indices[, axis, is_copy])</code>	Return the elements in the given <i>positional</i> indices along an axis.
<code>to_clipboard([excel, sep])</code>	Copy object to the system clipboard.
<code>to_csv([path_or_buf, sep, na_rep, ...])</code>	Write object to a comma-separated values (csv) file.
<code>to_dense()</code>	Return dense representation of Series/DataFrame (as opposed to sparse).
<code>to_dict([orient, into])</code>	Convert the DataFrame to a dictionary.
<code>to_excel(excel_writer[, sheet_name, na_rep, ...])</code>	Write object to an Excel sheet.
<code>to_feather(fname)</code>	Write out the binary feather-format for DataFrames.
<code>to_gbq(destination_table[, project_id, ...])</code>	Write a DataFrame to a Google BigQuery table.

Continued on next page

Table 10 – continued from previous page

<code>to_hdf(path_or_buf, key, **kwargs)</code>	Write the contained data to an HDF5 file using HDF-Store.
<code>to_html([buf, columns, col_space, header, ...])</code>	Render a DataFrame as an HTML table.
<code>to_json([path_or_buf, orient, date_format, ...])</code>	Convert the object to a JSON string.
<code>to_latex([buf, columns, col_space, header, ...])</code>	Render an object to a LaTeX tabular environment table.
<code>to_msgpack([path_or_buf, encoding])</code>	Serialize object to input file path using msgpack format.
<code>to_numpy([dtype, copy])</code>	Convert the DataFrame to a NumPy array.
<code>to_parquet(fname[, engine, compression, ...])</code>	Write a DataFrame to the binary parquet format.
<code>to_period([freq, axis, copy])</code>	Convert DataFrame from DatetimeIndex to PeriodIndex with desired frequency (inferred from index if not passed).
<code>to_pickle(path[, compression, protocol])</code>	Pickle (serialize) object to file.
<code>to_records([index, convert_datetime64, ...])</code>	Convert DataFrame to a NumPy record array.
<code>to_sparse([fill_value, kind])</code>	Convert to SparseDataFrame.
<code>to_sql(name, con[, schema, if_exists, ...])</code>	Write records stored in a DataFrame to a SQL database.
<code>to_stata(fname[, convert_dates, ...])</code>	Export DataFrame object to Stata dta format.
<code>to_string([buf, columns, col_space, header, ...])</code>	Render a DataFrame to a console-friendly tabular output.
<code>to_timestamp([freq, how, axis, copy])</code>	Cast to DatetimeIndex of timestamps, at <i>beginning</i> of period.
<code>to_xarray()</code>	Return an xarray object from the pandas object.
<code>transform(func[, axis])</code>	Call <code>func</code> on self producing a DataFrame with transformed values and that has the same axis length as self.
<code>transpose(*args, **kwargs)</code>	Transpose index and columns.
<code>truediv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>truncate([before, after, axis, copy])</code>	Truncate a Series or DataFrame before and after some index value.
<code>tshift([periods, freq, axis])</code>	Shift the time index, using the index's frequency if available.
<code>tz_convert(tz[, axis, level, copy])</code>	Convert tz-aware axis to target time zone.
<code>tz_localize(tz[, axis, level, copy, ...])</code>	Localize tz-naive index of a Series or DataFrame to target time zone.
<code>unstack([level, fill_value])</code>	Pivot a level of the (necessarily hierarchical) index labels, returning a DataFrame having a new level of column labels whose inner-most level consists of the pivoted index labels.
<code>update(other[, join, overwrite, ...])</code>	Modify in place using non-NA values from another DataFrame.
<code>var()</code>	Weighted variance of the sampled distribution.
<code>where(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is False.
<code>xs(key[, axis, level, drop_level])</code>	Return cross-section from the Series/DataFrame.

D (*nsamples=None*)

Kullback-Leibler divergence.

- If `nsamples` is not supplied, return mean KL divergence
- If `nsamples` is integer, return `nsamples` from the distribution

- If nsamples is array, use nsamples as volumes of evidence shells

beta

Thermodynamic inverse temperature.

d (*nsamples=None*)

Bayesian model dimensionality.

- If nsamples is not supplied, return mean BMD
- If nsamples is integer, return nsamples from the distribution
- If nsamples is array, use nsamples as volumes of evidence shells

dlogX (*nsamples=None*)

Compute volume of shell of loglikelihood.

Parameters

nsamples: int, optional Number of samples to generate. optional. If None, then compute the statistical average. If integer, generate samples from the distribution. (Default: None)

gui (*params=None*)

Construct a graphical user interface for viewing samples.

live_points (*logL=None*)

Get the live points within logL.

Parameters

logL: float or int, optional Loglikelihood or iteration number to return live points. If not provided, return the last set of active live points.

Returns

live_points: NestedSamples

Live points at either:

- contour logL (if input is float)
- ith contour (if input is integer)
- last generation contour if logL not provided

logZ (*nsamples=None*)

Log-Evidence.

- If nsamples is not supplied, return mean log evidence
- If nsamples is integer, return nsamples from the distribution
- If nsamples is array, use nsamples as volumes of evidence shells

ns_output (*nsamples=200*)

Compute Bayesian global quantities.

Using nested sampling we can compute the evidence (logZ), Kullback-Leibler divergence (D) and Bayesian model dimensionality (d). More precisely, we can infer these quantities via their probability distribution.

Parameters

nsamples: int, optional number of samples to generate (Default: 100)

Returns

pandas.DataFrame Samples from the $P(\log Z, D, d)$ distribution

posterior_points (*beta=1*)

Get equally weighted posterior points at temperature *beta*.

set_beta (*beta, inplace=False*)

Change the inverse temperature.

Parameters

beta: float Temperature to set

inplace: bool, optional Indicates whether to modify the existing array, or return a copy with the temperature changed. Default: False

`anesthetic.samples.merge_nested_samples` (*runs*)

Merge two or more nested sampling runs.

Parameters

runs: list(NestedSamples) list or array-like of nested sampling runs.

Returns

samples: NestedSamples Merged run.

7.6 anesthetic.utils module

Data-processing utility functions.

`anesthetic.utils.channel_capacity` (*w*)

Channel capacity (effective sample size).

$$H = \sum_i p_i \log p_i$$

$$p_i = \frac{w_i}{\sum_j w_j}$$

$$N = e^{-H}$$

`anesthetic.utils.check_bounds` (*d, xmin=None, xmax=None*)

Check if we need to apply strict bounds.

`anesthetic.utils.compress_weights` (*w, u=None, nsamples=None*)

Compresses weights to their approximate channel capacity.

`anesthetic.utils.compute_insertion_indexes` (*death, birth*)

Compute the live point insertion index for each point.

For more detail, see <https://arxiv.org/abs/2006.03371>

Parameters

death, birth [array-like] list of birth and death contours

Returns

indexes: np.array live point index at which each live point was inserted

`anesthetic.utils.compute_nlive` (*death, birth*)

Compute number of live points from birth and death contours.

Parameters

death, birth [array-like] list of birth and death contours

Returns

nlive: np.array number of live points at each contour

`anesthetic.utils.histogram(a, **kwargs)`

Produce a histogram for path-based plotting.

This is a cheap histogram. Necessary if one wants to update the histogram dynamically, and redrawing and filling is very expensive.

This has the same arguments and keywords as `np.histogram`, but is normalised to 1.

`anesthetic.utils.insertion_p_value(indexes, nlive, batch=0)`

Compute the p-value from insertion indexes, assuming constant nlive.

Note that this function doesn't use `scipy.stats.kstest` as the latter assumes continuous distributions.

For more detail, see <https://arxiv.org/abs/2006.03371>

For a rolling test, you should provide the optional parameter `batch!=0`. In this case the test computes the p value on consecutive batches of size `nlive * batch`, selects the smallest one and adjusts for multiple comparisons using a Bonferroni correction.

Parameters

indexes: array-like list of insertion indexes, sorted by death contour

nlive: int number of live points

batch: float batch size in units of nlive for a rolling p-value

Returns

ks_result: dict

Kolmogorov-Smirnov test results: D: Kolmogorov-Smirnov statistic sample_size: sample size p-value: p-value # if batch != 0 iterations: bounds of batch with minimum p-value nbatches: the number of batches in total uncorrected p-value: p-value without Bonferroni correction

`anesthetic.utils.is_int(x)`

Test whether x is an integer.

`anesthetic.utils.iso_probability_contours(pdf, contours=[0.68, 0.95])`

Compute the iso-probability contour values.

`anesthetic.utils.iso_probability_contours_from_samples(pdf, contours=[0.68, 0.95], weights=None)`

Compute the iso-probability contour values.

`anesthetic.utils.logsumexp(a, axis=None, b=None, keepdims=False, return_sign=False)`

Compute the log of the sum of exponentials of input elements.

This function has the same call signature as `scipy.special.logsumexp` and mirrors `scipy`'s behaviour except for `-np.inf` input. If a and b are both `-inf` then `scipy`'s function will output `nan` whereas here we use:

$$\lim_{x \rightarrow -\infty} x \exp(x) = 0$$

Thus, if `a=-inf` in `log(sum(b * exp(a)))` then we can set `b=0` such that that term is ignored in the sum.

`anesthetic.utils.match_contour_to_contourf(contours, vmin, vmax)`

Get needed `vmin, vmax` to match `contour` colors to `contourf` colors.

contourf uses the arithmetic mean of contour levels to assign colors, whereas *contour* uses the contour level directly. To get the same colors for *contour* lines as for *contourf* faces, we need some fiddly algebra.

`anesthetic.utils.mirror_1d` (*d*, *xmin=None*, *xmax=None*)

If necessary apply reflecting boundary conditions.

`anesthetic.utils.mirror_2d` (*d_x*, *d_y*, *xmin=None*, *xmax=None*, *ymin=None*, *ymax=None*)

If necessary apply reflecting boundary conditions.

`anesthetic.utils.nest_level` (*lst*)

Calculate the nesting level of a list.

`anesthetic.utils.quantile` (*a*, *q*, *w=None*)

Compute the weighted quantile for a one dimensional array.

`anesthetic.utils.sample_compression_1d` (*x*, *w=None*, *n=1000*)

Histogram a 1D set of weighted samples via subsampling.

This compresses the number of samples, combining weights.

Parameters

x: array-like x coordinate of samples for compressing

w: pandas.Series, optional weights of samples

n: int, optional number of samples returned. Default 1000

Returns

x, w, array-like Compressed samples and weights

`anesthetic.utils.scaled_triangulation` (*x*, *y*, *cov*)

Triangulation scaled by a covariance matrix.

Parameters

x, y: array-like x and y coordinates of samples

cov: array-like, 2d Covariance matrix for scaling

Returns

matplotlib.tri.Triangulation Triangulation with the appropriate scaling

`anesthetic.utils.temporary_seed` (*seed*)

Context for temporarily setting a numpy seed.

`anesthetic.utils.triangular_sample_compression_2d` (*x*, *y*, *cov*, *w=None*, *n=1000*)

Histogram a 2D set of weighted samples via triangulation.

This defines bins via a triangulation of the subsamples and sums weights within triangles surrounding each point

Parameters

x, y: array-like x and y coordinates of samples for compressing

cov: array-like, 2d Covariance matrix for scaling

w: pandas.Series, optional weights of samples

n: int, optional number of samples returned. Default 1000

Returns

tri: matplotlib.tri.Triangulation with an appropriate scaling

w: array-like Compressed samples and weights

`anesthetic.utils.unique(a)`
 Find unique elements, retaining order.

7.7 anesthetic.weighted_pandas module

Pandas DataFrame and Series with weighted samples.

class `anesthetic.weighted_pandas.WeightedDataFrame(*args, **kwargs)`
 Bases: `anesthetic.weighted_pandas._WeightedObject`, `pandas.core.frame.DataFrame`

Weighted version of `pandas.DataFrame`.

Attributes

- T** Transpose index and columns.
- at** Access a single value for a row/column label pair.
- axes** Return a list representing the axes of the DataFrame.
- blocks** Internal property, property synonym for `as_blocks()`.
- columns** The column labels of the DataFrame.
- dtypes** Return the dtypes in the DataFrame.
- empty** Indicator whether DataFrame is empty.
- ftypes** Return the ftypes (indication of sparse/dense and dtype) in DataFrame.
- iat** Access a single value for a row/column pair by integer position.
- iloc** Purely integer-location based indexing for selection by position.
- index** The index (row labels) of the DataFrame.
- is_copy** Return the copy.
- ix** A primarily label-location based indexer, with integer position fallback.
- loc** Access a group of rows and columns by label(s) or a boolean array.
- ndim** Return an int representing the number of axes / array dimensions.
- shape** Return a tuple representing the dimensionality of the DataFrame.
- size** Return an int representing the number of elements in this object.
- style** Property returning a Styler object containing methods for building a styled HTML representation fo the DataFrame.
- values** Return a Numpy representation of the DataFrame.
- weights** Sample weights.

Methods

<code>abs()</code>	Return a Series/DataFrame with absolute numeric value of each element.
--------------------	------------------------------------------------------------------------

Continued on next page

Table 11 – continued from previous page

<code>add(other[, axis, level, fill_value])</code>	Get Addition of dataframe and other, element-wise (binary operator <i>add</i>).
<code>add_prefix(prefix)</code>	Prefix labels with string <i>prefix</i> .
<code>add_suffix(suffix)</code>	Suffix labels with string <i>suffix</i> .
<code>agg(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>aggregate(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>align(other[, join, axis, level, copy, ...])</code>	Align two objects on their axes with the specified join method for each axis Index.
<code>all([axis, bool_only, skipna, level])</code>	Return whether all elements are True, potentially over an axis.
<code>any([axis, bool_only, skipna, level])</code>	Return whether any element is True, potentially over an axis.
<code>append(other[, ignore_index, ...])</code>	Append rows of <i>other</i> to the end of caller, returning a new object.
<code>apply(func[, axis, broadcast, raw, reduce, ...])</code>	Apply a function along an axis of the DataFrame.
<code>applymap(func)</code>	Apply a function to a DataFrame elementwise.
<code>as_blocks([copy])</code>	Convert the frame to a dict of dtype -> Constructor Types that each has a homogeneous dtype.
<code>as_matrix([columns])</code>	Convert the frame to its Numpy-array representation.
<code>asfreq(freq[, method, how, normalize, ...])</code>	Convert TimeSeries to specified frequency.
<code>asof(where[, subset])</code>	Return the last row(s) without any NaNs before <i>where</i> .
<code>assign(**kwargs)</code>	Assign new columns to a DataFrame.
<code>astype(dtype[, copy, errors])</code>	Cast a pandas object to a specified dtype <i>dtype</i> .
<code>at_time(time[, asof, axis])</code>	Select values at particular time of day (e.g.
<code>between_time(start_time, end_time[, ...])</code>	Select values between particular times of the day (e.g., 9:00-9:30 AM).
<code>bfill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='bfill'</code> .
<code>bool()</code>	Return the bool of a single element PandasObject.
<code>boxplot([column, by, ax, fontsize, rot, ...])</code>	Make a box plot from DataFrame columns.
<code>clip([lower, upper, axis, inplace])</code>	Trim values at input threshold(s).
<code>clip_lower(threshold[, axis, inplace])</code>	Trim values below a given threshold.
<code>clip_upper(threshold[, axis, inplace])</code>	Trim values above a given threshold.
<code>combine(other, func[, fill_value, overwrite])</code>	Perform column-wise combine with another DataFrame.
<code>combine_first(other)</code>	Update null elements with value in the same location in <i>other</i> .
<code>compound([axis, skipna, level])</code>	Return the compound percentage of the values for the requested axis.
<code>compress([nsamples])</code>	Reduce the number of samples by discarding low-weights.
<code>copy([deep])</code>	Make a copy of this object's indices and data.
<code>corr([method, min_periods])</code>	Compute pairwise correlation of columns, excluding NA/null values.
<code>corrwith(other[, axis, drop, method])</code>	Compute pairwise correlation between rows or columns of DataFrame with rows or columns of Series or DataFrame.
<code>count([axis, level, numeric_only])</code>	Count non-NA cells for each column or row.

Continued on next page

Table 11 – continued from previous page

<code>cov()</code>	Weighted covariance of the sampled distribution.
<code>cummax([axis, skipna])</code>	Return cumulative maximum over a DataFrame or Series axis.
<code>cummin([axis, skipna])</code>	Return cumulative minimum over a DataFrame or Series axis.
<code>cumprod([axis, skipna])</code>	Return cumulative product over a DataFrame or Series axis.
<code>cumsum([axis, skipna])</code>	Return cumulative sum over a DataFrame or Series axis.
<code>describe([percentiles, include, exclude])</code>	Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.
<code>diff([periods, axis])</code>	First discrete difference of element.
<code>div(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>divide(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>dot(other)</code>	Compute the matrix multiplication between the DataFrame and other.
<code>drop([labels, axis, index, columns, level, ...])</code>	Drop specified labels from rows or columns.
<code>drop_duplicates([subset, keep, inplace])</code>	Return DataFrame with duplicate rows removed, optionally only considering certain columns.
<code>droplevel(level[, axis])</code>	Return DataFrame with requested index / column level(s) removed.
<code>dropna([axis, how, thresh, subset, inplace])</code>	Remove missing values.
<code>duplicated([subset, keep])</code>	Return boolean Series denoting duplicate rows, optionally only considering certain columns.
<code>eq(other[, axis, level])</code>	Get Equal to of dataframe and other, element-wise (binary operator <i>eq</i>).
<code>equals(other)</code>	Test whether two objects contain the same elements.
<code>eval(expr[, inplace])</code>	Evaluate a string describing operations on DataFrame columns.
<code>ewm([com, span, halflife, alpha, ...])</code>	Provide exponential weighted functions.
<code>expanding([min_periods, center, axis])</code>	Provide expanding transformations.
<code>explode(column, Tuple)</code>	Transform each element of a list-like to a row, replicating the index values.
<code>ffill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='ffill'</code> .
<code>fillna([value, method, axis, inplace, ...])</code>	Fill NA/NaN values using the specified method.
<code>filter([items, like, regex, axis])</code>	Subset rows or columns of dataframe according to labels in the specified index.
<code>first(offset)</code>	Convenience method for subsetting initial periods of time series data based on a date offset.
<code>first_valid_index()</code>	Return index for first non-NA/null value.
<code>floordiv(other[, axis, level, fill_value])</code>	Get Integer division of dataframe and other, element-wise (binary operator <i>floordiv</i>).
<code>from_dict(data[, orient, dtype, columns])</code>	Construct DataFrame from dict of array-like or dicts.
<code>from_items(items[, columns, orient])</code>	Construct a DataFrame from a list of tuples.
<code>from_records(data[, index, exclude, ...])</code>	Convert structured or record ndarray to DataFrame.
<code>ge(other[, axis, level])</code>	Get Greater than or equal to of dataframe and other, element-wise (binary operator <i>ge</i>).

Continued on next page

Table 11 – continued from previous page

<code>get(key[, default])</code>	Get item from object for given key (ex: DataFrame column).
<code>get_dtype_counts()</code>	Return counts of unique dtypes in this object.
<code>get_ftype_counts()</code>	Return counts of unique ftypes in this object.
<code>get_value(index, col[, takeable])</code>	Quickly retrieve single value at passed column and index.
<code>get_values()</code>	Return an ndarray after converting sparse values to dense.
<code>groupby([by, axis, level, as_index, sort, ...])</code>	Group DataFrame or Series using a mapper or by a Series of columns.
<code>gt(other[, axis, level])</code>	Get Greater than of dataframe and other, element-wise (binary operator <i>gt</i>).
<code>head([n])</code>	Return the first <i>n</i> rows.
<code>hist(*args, **kwargs)</code>	Weighted histogram of the sampled distribution.
<code>idxmax([axis, skipna])</code>	Return index of first occurrence of maximum over requested axis.
<code>idxmin([axis, skipna])</code>	Return index of first occurrence of minimum over requested axis.
<code>infer_objects()</code>	Attempt to infer better dtypes for object columns.
<code>info([verbose, buf, max_cols, memory_usage, ...])</code>	Print a concise summary of a DataFrame.
<code>insert(loc, column, value[, allow_duplicates])</code>	Insert column into DataFrame at specified location.
<code>interpolate([method, axis, limit, inplace, ...])</code>	Interpolate values according to different methods.
<code>isin(values)</code>	Whether each element in the DataFrame is contained in values.
<code>isna()</code>	Detect missing values.
<code>isnull()</code>	Detect missing values.
<code>items()</code>	Iterator over (column name, Series) pairs.
<code>iteritems()</code>	Iterator over (column name, Series) pairs.
<code>iterrows()</code>	Iterate over DataFrame rows as (index, Series) pairs.
<code>itertuples([index, name])</code>	Iterate over DataFrame rows as namedtuples.
<code>join(other[, on, how, lsuffix, rsuffix, sort])</code>	Join columns of another DataFrame.
<code>keys()</code>	Get the ‘info axis’ (see Indexing for more)
<code>kurt([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher’s definition of kurtosis (kurtosis of normal == 0.0).
<code>kurtosis([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher’s definition of kurtosis (kurtosis of normal == 0.0).
<code>last(offset)</code>	Convenience method for subsetting final periods of time series data based on a date offset.
<code>last_valid_index()</code>	Return index for last non-NA/null value.
<code>le(other[, axis, level])</code>	Get Less than or equal to of dataframe and other, element-wise (binary operator <i>le</i>).
<code>lookup(row_labels, col_labels)</code>	Label-based “fancy indexing” function for DataFrame.
<code>lt(other[, axis, level])</code>	Get Less than of dataframe and other, element-wise (binary operator <i>lt</i>).
<code>mad([axis, skipna, level])</code>	Return the mean absolute deviation of the values for the requested axis.
<code>mask(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is True.

Continued on next page

Table 11 – continued from previous page

<code>max([axis, skipna, level, numeric_only])</code>	Return the maximum of the values for the requested axis.
<code>mean()</code>	Weighted mean of the sampled distribution.
<code>median()</code>	Weighted median of the sampled distribution.
<code>melt([id_vars, value_vars, var_name, ...])</code>	Unpivot a DataFrame from wide format to long format, optionally leaving identifier variables set.
<code>memory_usage([index, deep])</code>	Return the memory usage of each column in bytes.
<code>merge(right[, how, on, left_on, right_on, ...])</code>	Merge DataFrame or named Series objects with a database-style join.
<code>min([axis, skipna, level, numeric_only])</code>	Return the minimum of the values for the requested axis.
<code>mod(other[, axis, level, fill_value])</code>	Get Modulo of dataframe and other, element-wise (binary operator <i>mod</i>).
<code>mode([axis, numeric_only, dropna])</code>	Get the mode(s) of each element along the selected axis.
<code>mul(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>mul</i>).
<code>multiply(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>mul</i>).
<code>ne(other[, axis, level])</code>	Get Not equal to of dataframe and other, element-wise (binary operator <i>ne</i>).
<code>neff()</code>	Effective number of samples.
<code>nlargest(n, columns[, keep])</code>	Return the first <i>n</i> rows ordered by <i>columns</i> in descending order.
<code>notna()</code>	Detect existing (non-missing) values.
<code>notnull()</code>	Detect existing (non-missing) values.
<code>nsmallest(n, columns[, keep])</code>	Return the first <i>n</i> rows ordered by <i>columns</i> in ascending order.
<code>nunique([axis, dropna])</code>	Count distinct observations over requested axis.
<code>pct_change([periods, fill_method, limit, freq])</code>	Percentage change between the current and a prior element.
<code>pipe(func, *args, **kwargs)</code>	Apply <code>func(self, *args, **kwargs)</code> .
<code>pivot([index, columns, values])</code>	Return reshaped DataFrame organized by given index / column values.
<code>pivot_table([values, index, columns, ...])</code>	Create a spreadsheet-style pivot table as a DataFrame.
<code>plot</code>	alias of <code>pandas.plotting._core.PlotAccessor</code>
<code>pop(item)</code>	Return item and drop from frame.
<code>pow(other[, axis, level, fill_value])</code>	Get Exponential power of dataframe and other, element-wise (binary operator <i>pow</i>).
<code>prod([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>product([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>quantile([q])</code>	Weighted quantile of the sampled distribution.
<code>query(expr[, inplace])</code>	Query the columns of a DataFrame with a boolean expression.
<code>radd(other[, axis, level, fill_value])</code>	Get Addition of dataframe and other, element-wise (binary operator <i>radd</i>).

Continued on next page

Table 11 – continued from previous page

<code>rank([axis, method, numeric_only, ...])</code>	Compute numerical data ranks (1 through n) along axis.
<code>rdiv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>rtruediv</i>).
<code>reindex([labels, index, columns, axis, ...])</code>	Conform DataFrame to new index with optional filling logic, placing NA/NaN in locations having no value in the previous index.
<code>reindex_like(other[, method, copy, limit, ...])</code>	Return an object with matching indices as other object.
<code>rename([mapper, index, columns, axis, copy, ...])</code>	Alter axes labels.
<code>rename_axis([mapper, index, columns, axis, ...])</code>	Set the name of the axis for the index or columns.
<code>reorder_levels(order[, axis])</code>	Rearrange index levels using input order.
<code>replace([to_replace, value, inplace, limit, ...])</code>	Replace values given in <i>to_replace</i> with <i>value</i> .
<code>resample(rule[, how, axis, fill_method, ...])</code>	Resample time-series data.
<code>reset_index([level, drop, inplace, ...])</code>	Reset the index, or a level of it.
<code>rfloordiv(other[, axis, level, fill_value])</code>	Get Integer division of dataframe and other, element-wise (binary operator <i>rfloordiv</i>).
<code>rmod(other[, axis, level, fill_value])</code>	Get Modulo of dataframe and other, element-wise (binary operator <i>rmod</i>).
<code>rmul(other[, axis, level, fill_value])</code>	Get Multiplication of dataframe and other, element-wise (binary operator <i>rmul</i>).
<code>rolling(window[, min_periods, center, ...])</code>	Provide rolling window calculations.
<code>round([decimals])</code>	Round a DataFrame to a variable number of decimal places.
<code>rpow(other[, axis, level, fill_value])</code>	Get Exponential power of dataframe and other, element-wise (binary operator <i>rpow</i>).
<code>rsub(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>rsub</i>).
<code>rtruediv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>rtruediv</i>).
<code>sample([n, frac, replace, weights, ...])</code>	Return a random sample of items from an axis of object.
<code>select_dtypes([include, exclude])</code>	Return a subset of the DataFrame's columns based on the column dtypes.
<code>sem([axis, skipna, level, ddof, numeric_only])</code>	Return unbiased standard error of the mean over requested axis.
<code>set_axis(labels[, axis, inplace])</code>	Assign desired index to given axis.
<code>set_index(keys[, drop, append, inplace, ...])</code>	Set the DataFrame index using existing columns.
<code>set_value(index, col, value[, takeable])</code>	Put single value at passed column and index.
<code>shift([periods, freq, axis, fill_value])</code>	Shift index by desired number of periods with an optional time <i>freq</i> .
<code>skew([axis, skipna, level, numeric_only])</code>	Return unbiased skew over requested axis Normalized by N-1.
<code>slice_shift([periods, axis])</code>	Equivalent to <i>shift</i> without copying data.
<code>sort_index([axis, level, ascending, ...])</code>	Sort object by labels (along an axis).
<code>sort_values(by[, axis, ascending, inplace, ...])</code>	Sort by the values along either axis.
<code>sparse</code>	alias of <code>pandas.core.arrays.sparse.SparseFrameAccessor</code>
<code>squeeze([axis])</code>	Squeeze 1 dimensional axis objects into scalars.
<code>stack([level, dropna])</code>	Stack the prescribed level(s) from columns to index.

Continued on next page

Table 11 – continued from previous page

<code>std()</code>	Weighted standard deviation of the sampled distribution.
<code>sub(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>sub</i>).
<code>subtract(other[, axis, level, fill_value])</code>	Get Subtraction of dataframe and other, element-wise (binary operator <i>sub</i>).
<code>sum([axis, skipna, level, numeric_only, ...])</code>	Return the sum of the values for the requested axis.
<code>swapaxes(axis1, axis2[, copy])</code>	Interchange axes and swap values axes appropriately.
<code>swaplevel([i, j, axis])</code>	Swap levels i and j in a MultiIndex on a particular axis.
<code>tail([n])</code>	Return the last <i>n</i> rows.
<code>take(indices[, axis, is_copy])</code>	Return the elements in the given <i>positional</i> indices along an axis.
<code>to_clipboard([excel, sep])</code>	Copy object to the system clipboard.
<code>to_csv([path_or_buf, sep, na_rep, ...])</code>	Write object to a comma-separated values (csv) file.
<code>to_dense()</code>	Return dense representation of Series/DataFrame (as opposed to sparse).
<code>to_dict([orient, into])</code>	Convert the DataFrame to a dictionary.
<code>to_excel(excel_writer[, sheet_name, na_rep, ...])</code>	Write object to an Excel sheet.
<code>to_feather(fname)</code>	Write out the binary feather-format for DataFrames.
<code>to_gbq(destination_table[, project_id, ...])</code>	Write a DataFrame to a Google BigQuery table.
<code>to_hdf(path_or_buf, key, **kwargs)</code>	Write the contained data to an HDF5 file using HDF-Store.
<code>to_html([buf, columns, col_space, header, ...])</code>	Render a DataFrame as an HTML table.
<code>to_json([path_or_buf, orient, date_format, ...])</code>	Convert the object to a JSON string.
<code>to_latex([buf, columns, col_space, header, ...])</code>	Render an object to a LaTeX tabular environment table.
<code>to_msgpack([path_or_buf, encoding])</code>	Serialize object to input file path using msgpack format.
<code>to_numpy([dtype, copy])</code>	Convert the DataFrame to a NumPy array.
<code>to_parquet(fname[, engine, compression, ...])</code>	Write a DataFrame to the binary parquet format.
<code>to_period([freq, axis, copy])</code>	Convert DataFrame from DatetimeIndex to PeriodIndex with desired frequency (inferred from index if not passed).
<code>to_pickle(path[, compression, protocol])</code>	Pickle (serialize) object to file.
<code>to_records([index, convert_datetime64, ...])</code>	Convert DataFrame to a NumPy record array.
<code>to_sparse([fill_value, kind])</code>	Convert to SparseDataFrame.
<code>to_sql(name, con[, schema, if_exists, ...])</code>	Write records stored in a DataFrame to a SQL database.
<code>to_stata(fname[, convert_dates, ...])</code>	Export DataFrame object to Stata dta format.
<code>to_string([buf, columns, col_space, header, ...])</code>	Render a DataFrame to a console-friendly tabular output.
<code>to_timestamp([freq, how, axis, copy])</code>	Cast to DatetimeIndex of timestamps, at <i>beginning</i> of period.
<code>to_xarray()</code>	Return an xarray object from the pandas object.
<code>transform(func[, axis])</code>	Call <i>func</i> on self producing a DataFrame with transformed values and that has the same axis length as self.
<code>transpose(*args, **kwargs)</code>	Transpose index and columns.

Continued on next page

Table 11 – continued from previous page

<code>truediv(other[, axis, level, fill_value])</code>	Get Floating division of dataframe and other, element-wise (binary operator <i>truediv</i>).
<code>truncate([before, after, axis, copy])</code>	Truncate a Series or DataFrame before and after some index value.
<code>tshift([periods, freq, axis])</code>	Shift the time index, using the index's frequency if available.
<code>tz_convert(tz[, axis, level, copy])</code>	Convert tz-aware axis to target time zone.
<code>tz_localize(tz[, axis, level, copy, ...])</code>	Localize tz-naive index of a Series or DataFrame to target time zone.
<code>unstack([level, fill_value])</code>	Pivot a level of the (necessarily hierarchical) index labels, returning a DataFrame having a new level of column labels whose inner-most level consists of the pivoted index labels.
<code>update(other[, join, overwrite, ...])</code>	Modify in place using non-NA values from another DataFrame.
<code>var()</code>	Weighted variance of the sampled distribution.
<code>where(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is False.
<code>xs(key[, axis, level, drop_level])</code>	Return cross-section from the Series/DataFrame.

compress (*nsamples=None*)

Reduce the number of samples by discarding low-weights.

Parameters

neff: int, optional effective number of samples after compression. If not supplied, then reduce to the channel capacity (theoretical optimum compression). If ≤ 0 , then compress so that all weights are unity.

cov ()

Weighted covariance of the sampled distribution.

hist (**args, **kwargs*)

Weighted histogram of the sampled distribution.

mean ()

Weighted mean of the sampled distribution.

quantile (*q=0.5*)

Weighted quantile of the sampled distribution.

var ()

Weighted variance of the sampled distribution.

class `anesthetic.weighted_pandas.WeightedSeries` (**args, **kwargs*)

Bases: `anesthetic.weighted_pandas._WeightedObject, pandas.core.series.Series`

Weighted version of `pandas.Series`.

Attributes

T Return the transpose, which is by definition self.

array The `ExtensionArray` of the data backing this Series or Index.

asobject Return object Series which contains boxed values.

at Access a single value for a row/column label pair.

axes Return a list of the row axis labels.

base Return the base object if the memory of the underlying data is shared.

blocks Internal property, property synonym for `as_blocks()`.

data Return the data pointer of the underlying data.

dtype Return the dtype object of the underlying data.

dtypes Return the dtype object of the underlying data.

empty

flags Return the `ndarray.flags` for the underlying data.

ftype Return if the data is `sparsedense`.

ftypes Return if the data is `sparsedense`.

hasnans Return if I have any nans; enables various perf speedups.

iat Access a single value for a row/column pair by integer position.

iloc Purely integer-location based indexing for selection by position.

imag Return `imag` value of vector.

index The index (axis labels) of the Series.

is_copy Return the copy.

is_monotonic Return boolean if values in the object are `monotonic_increasing`.

is_monotonic_decreasing Return boolean if values in the object are `monotonic_decreasing`.

is_monotonic_increasing Return boolean if values in the object are `monotonic_increasing`.

is_unique Return boolean if values in the object are unique.

itemsizes Return the size of the dtype of the item of the underlying data.

ix A primarily label-location based indexer, with integer position fallback.

loc Access a group of rows and columns by label(s) or a boolean array.

name Return name of the Series.

nbytes Return the number of bytes in the underlying data.

ndim Number of dimensions of the underlying data, by definition 1.

real Return the real value of vector.

shape Return a tuple of the shape of the underlying data.

size Return the number of elements in the underlying data.

strides Return the strides of the underlying data.

values Return Series as `ndarray` or `ndarray-like` depending on the dtype.

weights Sample weights.

Methods

<code>abs()</code>	Return a Series/DataFrame with absolute numeric value of each element.
<code>add(other[, level, fill_value, axis])</code>	Return Addition of series and other, element-wise (binary operator <i>add</i>).
<code>add_prefix(prefix)</code>	Prefix labels with string <i>prefix</i> .
<code>add_suffix(suffix)</code>	Suffix labels with string <i>suffix</i> .
<code>agg(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>aggregate(func[, axis])</code>	Aggregate using one or more operations over the specified axis.
<code>align(other[, join, axis, level, copy, ...])</code>	Align two objects on their axes with the specified join method for each axis Index.
<code>all([axis, bool_only, skipna, level])</code>	Return whether all elements are True, potentially over an axis.
<code>any([axis, bool_only, skipna, level])</code>	Return whether any element is True, potentially over an axis.
<code>append(to_append[, ignore_index, ...])</code>	Concatenate two or more Series.
<code>apply(func[, convert_dtype, args])</code>	Invoke function on values of Series.
<code>argmax([axis, skipna])</code>	Return the row label of the maximum value.
<code>argmin([axis, skipna])</code>	Return the row label of the minimum value.
<code>argsort([axis, kind, order])</code>	Override ndarray.argsort.
<code>as_blocks([copy])</code>	Convert the frame to a dict of dtype -> Constructor Types that each has a homogeneous dtype.
<code>as_matrix([columns])</code>	Convert the frame to its Numpy-array representation.
<code>asfreq(freq[, method, how, normalize, ...])</code>	Convert TimeSeries to specified frequency.
<code>asof(where[, subset])</code>	Return the last row(s) without any NaNs before <i>where</i> .
<code>astype(dtype[, copy, errors])</code>	Cast a pandas object to a specified dtype <i>dtype</i> .
<code>at_time(time[, asof, axis])</code>	Select values at particular time of day (e.g.
<code>autocorr([lag])</code>	Compute the lag-N autocorrelation.
<code>between(left, right[, inclusive])</code>	Return boolean Series equivalent to <code>left <= series <= right</code> .
<code>between_time(start_time, end_time[, ...])</code>	Select values between particular times of the day (e.g., 9:00-9:30 AM).
<code>bfill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='bfill'</code> .
<code>bool()</code>	Return the bool of a single element PandasObject.
<code>cat</code>	alias of <code>pandas.core.arrays.categorical.CategoricalAccessor</code>
<code>clip([lower, upper, axis, inplace])</code>	Trim values at input threshold(s).
<code>clip_lower(threshold[, axis, inplace])</code>	Trim values below a given threshold.
<code>clip_upper(threshold[, axis, inplace])</code>	Trim values above a given threshold.
<code>combine(other, func[, fill_value])</code>	Combine the Series with a Series or scalar according to <i>func</i> .
<code>combine_first(other)</code>	Combine Series values, choosing the calling Series's values first.
<code>compound([axis, skipna, level])</code>	Return the compound percentage of the values for the requested axis.
<code>compress([nsamples])</code>	Reduce the number of samples by discarding low-weights.
<code>copy([deep])</code>	Make a copy of this object's indices and data.

Continued on next page

Table 12 – continued from previous page

<code>corr(other[, method, min_periods])</code>	Compute correlation with <i>other</i> Series, excluding missing values.
<code>count([level])</code>	Return number of non-NA/null observations in the Series.
<code>cov(other[, min_periods])</code>	Compute covariance with Series, excluding missing values.
<code>cummax([axis, skipna])</code>	Return cumulative maximum over a DataFrame or Series axis.
<code>cummin([axis, skipna])</code>	Return cumulative minimum over a DataFrame or Series axis.
<code>cumprod([axis, skipna])</code>	Return cumulative product over a DataFrame or Series axis.
<code>cumsum([axis, skipna])</code>	Return cumulative sum over a DataFrame or Series axis.
<code>describe([percentiles, include, exclude])</code>	Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.
<code>diff([periods])</code>	First discrete difference of element.
<code>div(other[, level, fill_value, axis])</code>	Return Floating division of series and other, element-wise (binary operator <i>truediv</i>).
<code>divide(other[, level, fill_value, axis])</code>	Return Floating division of series and other, element-wise (binary operator <i>truediv</i>).
<code>divmod(other[, level, fill_value, axis])</code>	Return Integer division and modulo of series and other, element-wise (binary operator <i>divmod</i>).
<code>dot(other)</code>	Compute the dot product between the Series and the columns of other.
<code>drop([labels, axis, index, columns, level, ...])</code>	Return Series with specified index labels removed.
<code>drop_duplicates([keep, inplace])</code>	Return Series with duplicate values removed.
<code>droplevel(level[, axis])</code>	Return DataFrame with requested index / column level(s) removed.
<code>dropna([axis, inplace])</code>	Return a new Series with missing values removed.
<code>dt</code>	alias of <code>pandas.core.indexes.accessors.CombinedDatetimelikeProperties</code>
<code>duplicated([keep])</code>	Indicate duplicate Series values.
<code>eq(other[, level, fill_value, axis])</code>	Return Equal to of series and other, element-wise (binary operator <i>eq</i>).
<code>equals(other)</code>	Test whether two objects contain the same elements.
<code>ewm([com, span, halflife, alpha, ...])</code>	Provide exponential weighted functions.
<code>expanding([min_periods, center, axis])</code>	Provide expanding transformations.
<code>explode()</code>	Transform each element of a list-like to a row, replicating the index values.
<code>factorize([sort, na_sentinel])</code>	Encode the object as an enumerated type or categorical variable.
<code>ffill([axis, inplace, limit, downcast])</code>	Synonym for <code>DataFrame.fillna()</code> with <code>method='ffill'</code> .
<code>fillna([value, method, axis, inplace, ...])</code>	Fill NA/NaN values using the specified method.
<code>filter([items, like, regex, axis])</code>	Subset rows or columns of dataframe according to labels in the specified index.
<code>first(offset)</code>	Convenience method for subsetting initial periods of time series data based on a date offset.

Continued on next page

Table 12 – continued from previous page

<code>first_valid_index()</code>	Return index for first non-NA/null value.
<code>floordiv(other[, level, fill_value, axis])</code>	Return Integer division of series and other, element-wise (binary operator <i>floordiv</i>).
<code>from_array(arr[, index, name, dtype, copy, ...])</code>	Construct Series from array.
<code>ge(other[, level, fill_value, axis])</code>	Return Greater than or equal to of series and other, element-wise (binary operator <i>ge</i>).
<code>get(key[, default])</code>	Get item from object for given key (ex: DataFrame column).
<code>get_dtype_counts()</code>	Return counts of unique dtypes in this object.
<code>get_ftype_counts()</code>	Return counts of unique ftypes in this object.
<code>get_value(label[, takeable])</code>	Quickly retrieve single value at passed index label.
<code>get_values()</code>	Same as values (but handles sparseness conversions); is a view.
<code>groupby([by, axis, level, as_index, sort, ...])</code>	Group DataFrame or Series using a mapper or by a Series of columns.
<code>gt(other[, level, fill_value, axis])</code>	Return Greater than of series and other, element-wise (binary operator <i>gt</i>).
<code>head([n])</code>	Return the first <i>n</i> rows.
<code>hist(*args, **kwargs)</code>	Weighted histogram of the sampled distribution.
<code>idxmax([axis, skipna])</code>	Return the row label of the maximum value.
<code>idxmin([axis, skipna])</code>	Return the row label of the minimum value.
<code>infer_objects()</code>	Attempt to infer better dtypes for object columns.
<code>interpolate([method, axis, limit, inplace, ...])</code>	Interpolate values according to different methods.
<code>isin(values)</code>	Check whether <i>values</i> are contained in Series.
<code>isna()</code>	Detect missing values.
<code>isnull()</code>	Detect missing values.
<code>item()</code>	Return the first element of the underlying data as a python scalar.
<code>items()</code>	Lazily iterate over (index, value) tuples.
<code>iteritems()</code>	Lazily iterate over (index, value) tuples.
<code>keys()</code>	Return alias for index.
<code>kurt([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher's definition of kurtosis (kurtosis of normal == 0.0).
<code>kurtosis([axis, skipna, level, numeric_only])</code>	Return unbiased kurtosis over requested axis using Fisher's definition of kurtosis (kurtosis of normal == 0.0).
<code>last(offset)</code>	Convenience method for subsetting final periods of time series data based on a date offset.
<code>last_valid_index()</code>	Return index for last non-NA/null value.
<code>le(other[, level, fill_value, axis])</code>	Return Less than or equal to of series and other, element-wise (binary operator <i>le</i>).
<code>lt(other[, level, fill_value, axis])</code>	Return Less than of series and other, element-wise (binary operator <i>lt</i>).
<code>mad([axis, skipna, level])</code>	Return the mean absolute deviation of the values for the requested axis.
<code>map(arg[, na_action])</code>	Map values of Series according to input correspondence.
<code>mask(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is True.
<code>max([axis, skipna, level, numeric_only])</code>	Return the maximum of the values for the requested axis.

Continued on next page

Table 12 – continued from previous page

<code>mean()</code>	Weighted mean of the sampled distribution.
<code>median()</code>	Weighted median of the sampled distribution.
<code>memory_usage([index, deep])</code>	Return the memory usage of the Series.
<code>min([axis, skipna, level, numeric_only])</code>	Return the minimum of the values for the requested axis.
<code>mod(other[, level, fill_value, axis])</code>	Return Modulo of series and other, element-wise (binary operator <i>mod</i>).
<code>mode([dropna])</code>	Return the mode(s) of the dataset.
<code>mul(other[, level, fill_value, axis])</code>	Return Multiplication of series and other, element-wise (binary operator <i>mul</i>).
<code>multiply(other[, level, fill_value, axis])</code>	Return Multiplication of series and other, element-wise (binary operator <i>mul</i>).
<code>ne(other[, level, fill_value, axis])</code>	Return Not equal to of series and other, element-wise (binary operator <i>ne</i>).
<code>neff()</code>	Effective number of samples.
<code>nlargest([n, keep])</code>	Return the largest <i>n</i> elements.
<code>nonzero()</code>	Return the <i>integer</i> indices of the elements that are non-zero.
<code>notna()</code>	Detect existing (non-missing) values.
<code>notnull()</code>	Detect existing (non-missing) values.
<code>nsmallest([n, keep])</code>	Return the smallest <i>n</i> elements.
<code>nunique([dropna])</code>	Return number of unique elements in the object.
<code>pct_change([periods, fill_method, limit, freq])</code>	Percentage change between the current and a prior element.
<code>pipe(func, *args, **kwargs)</code>	Apply <code>func(self, *args, **kwargs)</code> .
<code>plot</code>	alias of <code>pandas.plotting._core.PlotAccessor</code>
<code>pop(item)</code>	Return item and drop from frame.
<code>pow(other[, level, fill_value, axis])</code>	Return Exponential power of series and other, element-wise (binary operator <i>pow</i>).
<code>prod([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>product([axis, skipna, level, numeric_only, ...])</code>	Return the product of the values for the requested axis.
<code>ptp([axis, skipna, level, numeric_only])</code>	Return the difference between the maximum value and the
<code>put(*args, **kwargs)</code>	Apply the <i>put</i> method to its <i>values</i> attribute if it has one.
<code>quantile([q])</code>	Weighted quantile of the sampled distribution.
<code>radd(other[, level, fill_value, axis])</code>	Return Addition of series and other, element-wise (binary operator <i>radd</i>).
<code>rank([axis, method, numeric_only, ...])</code>	Compute numerical data ranks (1 through n) along axis.
<code>ravel([order])</code>	Return the flattened underlying data as an ndarray.
<code>rdiv(other[, level, fill_value, axis])</code>	Return Floating division of series and other, element-wise (binary operator <i>rtruediv</i>).
<code>rdivmod(other[, level, fill_value, axis])</code>	Return Integer division and modulo of series and other, element-wise (binary operator <i>rdivmod</i>).
<code>reindex([index])</code>	Conform Series to new index with optional filling logic, placing NA/NaN in locations having no value in the previous index.

Continued on next page

Table 12 – continued from previous page

<code>reindex_like(other[, method, copy, limit, ...])</code>	Return an object with matching indices as other object.
<code>rename([index])</code>	Alter Series index labels or name.
<code>rename_axis([mapper, index, columns, axis, ...])</code>	Set the name of the axis for the index or columns.
<code>reorder_levels(order)</code>	Rearrange index levels using input order.
<code>repeat(repeats[, axis])</code>	Repeat elements of a Series.
<code>replace([to_replace, value, inplace, limit, ...])</code>	Replace values given in <i>to_replace</i> with <i>value</i> .
<code>resample(rule[, how, axis, fill_method, ...])</code>	Resample time-series data.
<code>reset_index([level, drop, name, inplace])</code>	Generate a new DataFrame or Series with the index reset.
<code>rfloordiv(other[, level, fill_value, axis])</code>	Return Integer division of series and other, element-wise (binary operator <i>rfloordiv</i>).
<code>rmod(other[, level, fill_value, axis])</code>	Return Modulo of series and other, element-wise (binary operator <i>rmod</i>).
<code>rmul(other[, level, fill_value, axis])</code>	Return Multiplication of series and other, element-wise (binary operator <i>rmul</i>).
<code>rolling(window[, min_periods, center, ...])</code>	Provide rolling window calculations.
<code>round([decimals])</code>	Round each value in a Series to the given number of decimals.
<code>rpow(other[, level, fill_value, axis])</code>	Return Exponential power of series and other, element-wise (binary operator <i>rpow</i>).
<code>rsub(other[, level, fill_value, axis])</code>	Return Subtraction of series and other, element-wise (binary operator <i>rsub</i>).
<code>rtruediv(other[, level, fill_value, axis])</code>	Return Floating division of series and other, element-wise (binary operator <i>rtruediv</i>).
<code>sample([n, frac, replace, weights, ...])</code>	Return a random sample of items from an axis of object.
<code>searchsorted(value[, side, sorter])</code>	Find indices where elements should be inserted to maintain order.
<code>sem([axis, skipna, level, ddof, numeric_only])</code>	Return unbiased standard error of the mean over requested axis.
<code>set_axis(labels[, axis, inplace])</code>	Assign desired index to given axis.
<code>set_value(label, value[, takeable])</code>	Quickly set single value at passed label.
<code>shift([periods, freq, axis, fill_value])</code>	Shift index by desired number of periods with an optional time <i>freq</i> .
<code>skew([axis, skipna, level, numeric_only])</code>	Return unbiased skew over requested axis Normalized by N-1.
<code>slice_shift([periods, axis])</code>	Equivalent to <i>shift</i> without copying data.
<code>sort_index([axis, level, ascending, ...])</code>	Sort Series by index labels.
<code>sort_values([axis, ascending, inplace, ...])</code>	Sort by the values.
<code>sparse</code>	alias of <code>pandas.core.arrays.sparse.SparseAccessor</code>
<code>squeeze([axis])</code>	Squeeze 1 dimensional axis objects into scalars.
<code>std()</code>	Weighted standard deviation of the sampled distribution.
<code>str</code>	alias of <code>pandas.core.strings.StringMethods</code>
<code>sub(other[, level, fill_value, axis])</code>	Return Subtraction of series and other, element-wise (binary operator <i>sub</i>).

Continued on next page

Table 12 – continued from previous page

<code>subtract(other[, level, fill_value, axis])</code>	Return Subtraction of series and other, element-wise (binary operator <i>sub</i>).
<code>sum([axis, skipna, level, numeric_only, ...])</code>	Return the sum of the values for the requested axis.
<code>swapaxes(axis1, axis2[, copy])</code>	Interchange axes and swap values axes appropriately.
<code>swaplevel([i, j, copy])</code>	Swap levels i and j in a MultiIndex.
<code>tail([n])</code>	Return the last <i>n</i> rows.
<code>take(indices[, axis, is_copy])</code>	Return the elements in the given <i>positional</i> indices along an axis.
<code>to_clipboard([excel, sep])</code>	Copy object to the system clipboard.
<code>to_csv(*args, **kwargs)</code>	Write object to a comma-separated values (csv) file.
<code>to_dense()</code>	Return dense representation of Series/DataFrame (as opposed to sparse).
<code>to_dict([into])</code>	Convert Series to {label -> value} dict or dict-like object.
<code>to_excel(excel_writer[, sheet_name, na_rep, ...])</code>	Write object to an Excel sheet.
<code>to_frame([name])</code>	Convert Series to DataFrame.
<code>to_hdf(path_or_buf, key, **kwargs)</code>	Write the contained data to an HDF5 file using HDF-Store.
<code>to_json([path_or_buf, orient, date_format, ...])</code>	Convert the object to a JSON string.
<code>to_latex([buf, columns, col_space, header, ...])</code>	Render an object to a LaTeX tabular environment table.
<code>to_list()</code>	Return a list of the values.
<code>to_msgpack([path_or_buf, encoding])</code>	Serialize object to input file path using msgpack format.
<code>to_numpy([dtype, copy])</code>	A NumPy ndarray representing the values in this Series or Index.
<code>to_period([freq, copy])</code>	Convert Series from DatetimeIndex to PeriodIndex with desired frequency (inferred from index if not passed).
<code>to_pickle(path[, compression, protocol])</code>	Pickle (serialize) object to file.
<code>to_sparse([kind, fill_value])</code>	Convert Series to SparseSeries.
<code>to_sql(name, con[, schema, if_exists, ...])</code>	Write records stored in a DataFrame to a SQL database.
<code>to_string([buf, na_rep, float_format, ...])</code>	Render a string representation of the Series.
<code>to_timestamp([freq, how, copy])</code>	Cast to DatetimeIndex of Timestamps, at <i>beginning</i> of period.
<code>to_xarray()</code>	Return an xarray object from the pandas object.
<code>tolist()</code>	Return a list of the values.
<code>transform(func[, axis])</code>	Call <i>func</i> on self producing a Series with transformed values and that has the same axis length as self.
<code>transpose(*args, **kwargs)</code>	Return the transpose, which is by definition self.
<code>truediv(other[, level, fill_value, axis])</code>	Return Floating division of series and other, element-wise (binary operator <i>truediv</i>).
<code>truncate([before, after, axis, copy])</code>	Truncate a Series or DataFrame before and after some index value.
<code>tshift([periods, freq, axis])</code>	Shift the time index, using the index's frequency if available.
<code>tz_convert(tz[, axis, level, copy])</code>	Convert tz-aware axis to target time zone.

Continued on next page

Table 12 – continued from previous page

<code>tz_localize(tz[, axis, level, copy, ...])</code>	Localize tz-naive index of a Series or DataFrame to target time zone.
<code>unique()</code>	Return unique values of Series object.
<code>unstack([level, fill_value])</code>	Unstack, a.k.a.
<code>update(other)</code>	Modify Series in place using non-NA values from passed Series.
<code>valid([inplace])</code>	Return Series without null values.
<code>value_counts([normalize, sort, ascending, ...])</code>	Return a Series containing counts of unique values.
<code>var()</code>	Weighted variance of the sampled distribution.
<code>view([dtype])</code>	Create a new view of the Series.
<code>where(cond[, other, inplace, axis, level, ...])</code>	Replace values where the condition is False.
<code>xs(key[, axis, level, drop_level])</code>	Return cross-section from the Series/DataFrame.

compress (*nsamples=None*)

Reduce the number of samples by discarding low-weights.

Parameters

neff: int, optional effective number of samples after compression. If not supplied, then reduce to the channel capacity (theoretical optimum compression). If ≤ 0 , then compress so that all weights are unity.

hist (**args, **kwargs*)

Weighted histogram of the sampled distribution.

mean ()

Weighted mean of the sampled distribution.

quantile (*q=0.5*)

Weighted quantile of the sampled distribution.

var ()

Weighted variance of the sampled distribution.

7.8 Module contents

Anesthetic: nested sampling post-processing.

Key routines:

- `MCMCSamples.build`
- `MCMCSamples.read`
- `NestedSamples.build`
- `NestedSamples.read`

anesthetic: nested sampling visualisation

anesthetic nested sampling visualisation

Author Will Handley and Lukas Hergt

Version 2.0.0-beta.1

Homepage <https://github.com/williamjameshandley/anesthetic>

Documentation <http://anesthetic.readthedocs.io/>

pypi package 1.3.6

pypi package 1.3.6

codecov 91%

docs failing

pypi package 1.3.6

DOI 10.5281/zenodo.3968183

JOSS 10.21105/joss.01414

license MIT

 **launch binder**

`anesthetic` brings together tools for processing nested sampling chains, leveraging standard scientific python libraries.

You can see example usage and plots in the [plot gallery](#), or in the corresponding [Jupyter notebook](#).

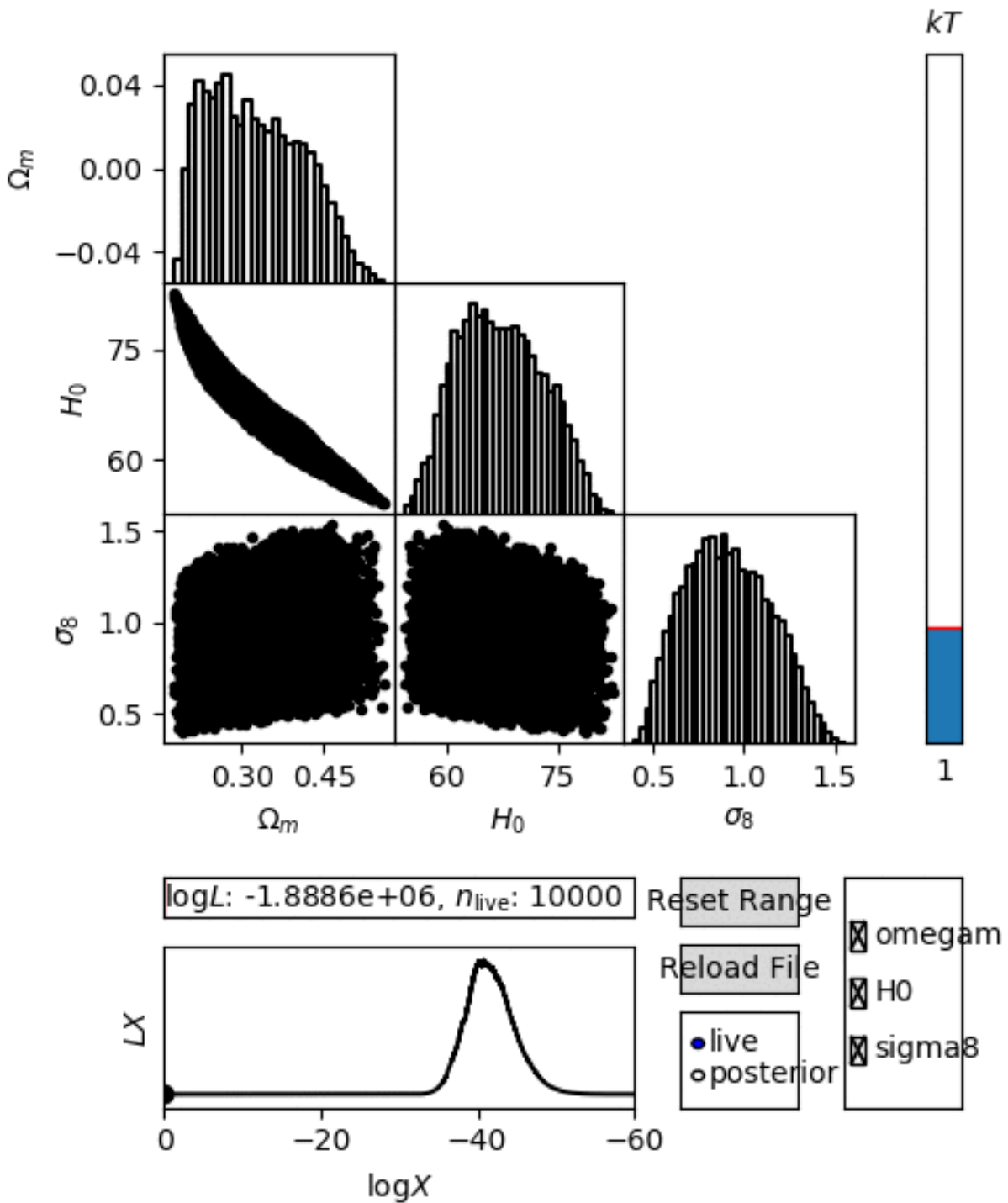
Current functionality includes:

- Computation of Bayesian evidences, Kullback-Liebler divergences and Bayesian model dimensionalities.
- Marginalised 1d and 2d plots.
- Dynamic replaying of nested sampling.

This tool was designed primarily for use with nested sampling outputs, although it can be used for normal MCMC chains.

For an interactive view of a nested sampling run, you can use the `anesthetic` script.

```
$ anesthetic <ns file root>
```



8.1 Features

- Both samples and plotting axes are stored as a `pandas.DataFrame`, with parameter names as indices, which makes for easy access and modification.

- Sensible color scheme for plotting nearly flat distributions.
- For easy extension/modification, uses the standard python libraries: `numpy`, `scipy`, `matplotlib` and `pandas`.

8.2 Installation

`anesthetic` can be installed via `pip`

```
pip install anesthetic
```

or via the `setup.py`

```
git clone https://github.com/williamjameshandley/anesthetic
cd anesthetic
python setup.py install --user
```

You can check that things are working by running the test suite:

```
export MPLBACKEND=Agg      # only necessary for OSX users
python -m pytest
flake8 anesthetic tests
pydocstyle --convention=numpy anesthetic
```

8.2.1 Dependencies

Basic requirements:

- Python 3.6+
- `matplotlib`
- `numpy`
- `scipy`
- `pandas`
- `fastKDE`

Documentation:

- `sphinx`
- `numpydoc`

Tests:

- `pytest`

8.3 Documentation

Full Documentation is hosted at [ReadTheDocs](#). To build your own local copy of the documentation you'll need to install `sphinx`. You can then run:

```
cd docs
make html
```

8.4 Citation

If you use `anesthetic` to generate plots for a publication, please cite as:

Handley, (2019). `anesthetic`: nested sampling visualisation. Journal of Open Source Software, 4(37), 1414, <https://doi.org/10.21105/joss.01414>

or using the BibTeX:

```
@article{anesthetic,
  doi = {10.21105/joss.01414},
  url = {http://dx.doi.org/10.21105/joss.01414},
  year = {2019},
  month = {Jun},
  publisher = {The Open Journal},
  volume = {4},
  number = {37},
  pages = {1414},
  author = {Will Handley},
  title = {anesthetic: nested sampling visualisation},
  journal = {The Journal of Open Source Software}
}
```

8.5 Contributing

There are many ways you can contribute via the [GitHub repository](#).

- You can [open an issue](#) to report bugs or to propose new features.
- Pull requests are very welcome. Note that if you are going to propose major changes, be sure to open an issue for discussion first, to make sure that your PR will be accepted before you spend effort coding it.

8.6 Questions/Comments

8.6.1 Another posterior plotting tool?

This is my posterior plotter. There are many like it, but this one is mine.

There are several excellent tools for plotting marginalised posteriors:

- [getdist](#)
- [corner](#)
- [pygtc](#)
- [dynesty](#)
- [MontePython](#)

Why create another one? In general, any dedicated user of software will find that there is some functionality that in their use case is lacking, and the designs of previous codes make such extensions challenging. In my case this was:

1. For large numbers of samples, kernel density estimation is slow, or inaccurate (particularly for samples generated from nested sampling). There are kernel density estimators, such as [fastKDE](#), which ameliorate many of these difficulties.

- Existing tools can make it difficult to define new parameters. For example, the default cosmomc chain defines `omegabh2`, but not `omegab`. The transformation is easy, since $\text{omegab} = \text{omegab}_{h2} / (H_0/100)^{**2}$, but implementing this transformation in existing packages is not so trivial. `anesthetic` solves this issue by storing the samples as a pandas array, for which the relevant code for defining the above new parameter would be

```
from anesthetic import MCMCSamples
samples = MCMCSamples(root=file_root)           # Load the samples
samples['omegab'] = samples.omegabh2 / (samples.H0/100)**2 # Define omegab
samples.tex['omegab'] = '$\Omega_b$'           # Label omegab
samples.plot_1d('omegab')                     # Simple 1D plot
```

- Many KDE plotting tools have conventions that don't play well with uniformly distributed parameters, which presents a problem if you are trying to plot priors along with your posteriors. `anesthetic` has a sensible mechanism, by defining the contours by the amount of iso-probability mass they contain, but colouring the fill in relation to the probability density of the contour.

8.6.2 What's in a name?

There is an emerging convention for naming nested sampling packages with words that have nest in them (`nestle` and `dynesty`, `nestorflow`). Doing a UNIX `grep`:

```
grep nest /usr/share/dict/words
```

yields a lot of superlatives (e.g. `greenest`), but a few other cool names for future projects:

- `amnesty`
- `defenestrate`
- `dishonestly`
- `inestimable`
- `minestrone`
- `rhinestone`

I chose `anesthetic` because I liked the soft 'th', and in spite of the US spelling.

a

anesthetic, 62
anesthetic.gui, 21
anesthetic.gui.plot, 15
anesthetic.gui.widgets, 18
anesthetic.plot, 21
anesthetic.read, 21
anesthetic.samples, 25
anesthetic.utils, 44
anesthetic.weighted_pandas, 47

A

anesthetic (*module*), 62
 anesthetic.gui (*module*), 21
 anesthetic.gui.plot (*module*), 15
 anesthetic.gui.widgets (*module*), 18
 anesthetic.plot (*module*), 21
 anesthetic.read (*module*), 21
 anesthetic.samples (*module*), 25
 anesthetic.utils (*module*), 44
 anesthetic.weighted_pandas (*module*), 47

B

basic_cmap() (*in module anesthetic.plot*), 21
 beta (*anesthetic.samples.NestedSamples attribute*), 43
 Button (*class in anesthetic.gui.widgets*), 18

C

channel_capacity() (*in module anesthetic.utils*), 44
 check_bounds() (*in module anesthetic.utils*), 44
 CheckButtons (*class in anesthetic.gui.widgets*), 18
 compress() (*anesthetic.weighted_pandas.WeightedDataFrame method*), 54
 compress() (*anesthetic.weighted_pandas.WeightedSeries method*), 62
 compress_weights() (*in module anesthetic.utils*), 44
 compute_insertion_indexes() (*in module anesthetic.utils*), 44
 compute_nlive() (*in module anesthetic.utils*), 44
 cov() (*anesthetic.weighted_pandas.WeightedDataFrame method*), 54

D

D() (*anesthetic.samples.NestedSamples method*), 42
 d() (*anesthetic.samples.NestedSamples method*), 43
 dlogX() (*anesthetic.samples.NestedSamples method*), 43

draw() (*anesthetic.gui.widgets.TrianglePlot method*), 20

E

Evolution (*class in anesthetic.gui.plot*), 15

F

fastkde_contour_plot_2d() (*in module anesthetic.plot*), 21
 fastkde_plot_1d() (*in module anesthetic.plot*), 22

G

get_legend_proxy() (*in module anesthetic.plot*), 22
 gui() (*anesthetic.samples.NestedSamples method*), 43

H

Higson (*class in anesthetic.gui.plot*), 15
 hist() (*anesthetic.weighted_pandas.WeightedDataFrame method*), 54
 hist() (*anesthetic.weighted_pandas.WeightedSeries method*), 62
 hist_plot_1d() (*in module anesthetic.plot*), 22
 hist_plot_2d() (*in module anesthetic.plot*), 23
 histogram() (*in module anesthetic.utils*), 45

I

insertion_p_value() (*in module anesthetic.utils*), 45
 is_int() (*in module anesthetic.utils*), 45
 iso_probability_contours() (*in module anesthetic.utils*), 45
 iso_probability_contours_from_samples() (*in module anesthetic.utils*), 45

K

kde_contour_plot_2d() (*in module anesthetic.plot*), 23
 kde_plot_1d() (*in module anesthetic.plot*), 23

L

LabelsWidget (class in *anesthetic.gui.widgets*), 18
 live_points() (*anesthetic.samples.NestedSamples* method), 43
 logsumexp() (in module *anesthetic.utils*), 45
 logZ() (*anesthetic.samples.NestedSamples* method), 43

M

make_1d_axes() (in module *anesthetic.plot*), 24
 make_2d_axes() (in module *anesthetic.plot*), 24
 make_diagonal() (in module *anesthetic.plot*), 24
 match_contour_to_contourf() (in module *anesthetic.utils*), 45
 MCMCSamples (class in *anesthetic.samples*), 25
 mean() (*anesthetic.weighted_pandas.WeightedDataFrame* method), 54
 mean() (*anesthetic.weighted_pandas.WeightedSeries* method), 62
 merge_nested_samples() (in module *anesthetic.samples*), 44
 mirror_1d() (in module *anesthetic.utils*), 46
 mirror_2d() (in module *anesthetic.utils*), 46

N

nest_level() (in module *anesthetic.utils*), 46
 NestedSamples (class in *anesthetic.samples*), 35
 normalize_kwargs() (in module *anesthetic.plot*), 24
 ns_output() (*anesthetic.samples.NestedSamples* method), 43

P

plot() (*anesthetic.samples.MCMCSamples* method), 33
 plot_1d() (*anesthetic.samples.MCMCSamples* method), 33
 plot_2d() (*anesthetic.samples.MCMCSamples* method), 34
 points() (*anesthetic.gui.plot.RunPlotter* method), 17
 posterior_points() (*anesthetic.samples.NestedSamples* method), 44

Q

quantile() (*anesthetic.weighted_pandas.WeightedDataFrame* method), 54
 quantile() (*anesthetic.weighted_pandas.WeightedSeries* method), 62
 quantile() (in module *anesthetic.utils*), 46
 quantile_plot_interval() (in module *anesthetic.plot*), 25

R

RadioButtons (class in *anesthetic.gui.widgets*), 18

redraw() (*anesthetic.gui.plot.RunPlotter* method), 17
 reload_file() (*anesthetic.gui.plot.RunPlotter* method), 17
 reset_range() (*anesthetic.gui.plot.Higson* method), 16
 reset_range() (*anesthetic.gui.plot.RunPlotter* method), 17
 reset_range() (*anesthetic.gui.widgets.Slider* method), 19
 reset_range() (*anesthetic.gui.widgets.TrianglePlot* method), 20
 RunPlotter (class in *anesthetic.gui.plot*), 16

S

sample_compression_1d() (in module *anesthetic.utils*), 46
 scaled_triangulation() (in module *anesthetic.utils*), 46
 scatter_plot_2d() (in module *anesthetic.plot*), 25
 set_beta() (*anesthetic.samples.NestedSamples* method), 44
 set_text() (*anesthetic.gui.plot.Evolution* method), 15
 set_text() (*anesthetic.gui.plot.Temperature* method), 17
 set_text() (*anesthetic.gui.widgets.Slider* method), 19
 Slider (class in *anesthetic.gui.widgets*), 19

T

Temperature (class in *anesthetic.gui.plot*), 17
 temporary_seed() (in module *anesthetic.utils*), 46
 TrianglePlot (class in *anesthetic.gui.widgets*), 19
 triangular_sample_compression_2d() (in module *anesthetic.utils*), 46

U

unique() (in module *anesthetic.utils*), 46
 update() (*anesthetic.gui.plot.Higson* method), 16
 update() (*anesthetic.gui.plot.RunPlotter* method), 17
 update() (*anesthetic.gui.widgets.TrianglePlot* method), 20

V

var() (*anesthetic.weighted_pandas.WeightedDataFrame* method), 54
 var() (*anesthetic.weighted_pandas.WeightedSeries* method), 62

W

WeightedDataFrame (class in *anesthetic.weighted_pandas*), 47
 WeightedSeries (class in *anesthetic.weighted_pandas*), 54
 Widget (class in *anesthetic.gui.widgets*), 20