sciutils

Till Hoffmann

Jun 07, 2021

CONTENTS

A collection of functionality useful for scientific computing.

CHAPTER

ONE

UTILITY FUNCTIONS FOR PLOTTING

`sciutils.plot.`**`alpha_cmap`**(*color*, *name=''*)

Create a monochrome colormap that maps scalars to varying transparencies.

Parameters

- **`color`** (*`str, int, or tuple`*) – Base color to use for the colormap.
- **`name`** (*`str`*) – Name of the colormap.
- **`**kwargs`** (*`dict`*) – Keyword arguments passed to `mpl.colors.LinearSegmentedColormap.from_list()`.

Returns **cmap** – Colormap encoding scalars as transparencies.

Return type mpl.colors.Colormap

`sciutils.plot.`**`evaluate_pcolormesh_edges`**(*x*, *scale='linear'*)

Evaluate the *n + 1* edges of cells for a *pcolormesh* visualisation for *n* cell centroids.

Parameters

- **`x`** (*`np.ndarray`*) – Centroids of the pcolormesh cells.
- **`scale`** (*`str`*) – Find the arithmetic midpoints if *linear* and the geometric midpoints if *log*.

Returns **edges** – Edges of pcolormesh cells.

Return type np.ndarray

`sciutils.plot.`**`plot_geometry`**(*geometries*, *aspect='equal'*, *autoscale=True*, *scale=1*, *ax=None*, ***kwargs*)

Plot a shapely geometry using a polygon collection.

Note: This function does not plot holes in polygons.

Parameters

- **`geometries`** – Geometry to plot or sequence thereof.
- **`aspect`** (*`str or float, optional`*) – Aspect ratio of the plot.
- **`autoscale`** (*`bool, optional`*) – Whether to autoscale the plot.
- **`ax`** (*`optional`*) – Axes to use for plotting.
- **`**kwargs`** (*`dict`*) – Keyword arguments passed to *matplotlib.collections.PolyCollection*.

Returns **collection** – Collection of polygons.

Return type matplotlib.collections.PolyCollection

CHAPTER

TWO

STATISTICAL DISTRIBUTIONS AND UTILITY FUNCTIONS

Fast, numerically stable implementations of log PDFs and CDFs as well as statistical utility functions.

Note: Distributions are only guaranteed to be correct within their support. E.g. the behaviour of evaluating a Gamma distribution for negative values is undefined.

class sciutils.stats.**BoundedVariable**(*a=0*, *b=1*)

A bounded variable $y = a + \frac{(b-a)}{1+\exp(-x)}$ on the interval $[a, b]$.

apply(*x*)

Transform a variable from an unconstrained space to a possibly constrained space.

Parameters **x** (*array_like*) – Variable to transform.

Returns

- **y** (*array_like*) – Transformed variable.
- **log_jacobian** (*array_like*) – Logarithm of the Jacobian associated with the transform.

invert(*y*)

Transform a variable from a possibly constrained space to an untransformed space.

Parameters **y** (*array_like*) – Transformed variable.

Returns **x** – Variable after inverse transform.

Return type array_like

class sciutils.stats.**ParameterReshaper**(*parameters*)

Reshape an array of parameters to a dictionary of named parameters and vice versa.

Trailing dimensions of each parameter are considered batch dimensions and are left unchanged.

Parameters **parameters** (*dict[str, tuple]*) – Mapping from parameter names to shapes.

Examples

```
>>> reshaper = su.stats.ParameterReshaper({'a': 2, 'b': (2, 3)})
>>> reshaper.to_dict(np.arange(reshaper.size))
{'a': array([0, 1]), 'b': array([[2, 3, 4], [5, 6, 7]])}
```

to_array(*values*, *moveaxis=False*, *validate=True*)

Convert a dictionary of values to an array.

Parameters

- **values** (*dict[str, np.ndarray]*) – Mapping from parameter names to values.
- **moveaxis** (*bool*) – Move the first axis to the last dimension after reshaping to an array, e.g. if the batch dimensions are leading.
- **validate** (*bool*) – Validate the input at some cost to performance.

Returns **array** – Array of parameters encoding the named parameters.

Return type np.ndarray

to_dict (*array*, *moveaxis=False*, *validate=True*)

Convert an array to a dictionary of values.

Trailing dimensions of the array are considered batch dimensions and are left unchanged.

Parameters

- **array** (*np.ndarray*) – Array of parameters encoding a parameter set.
- **moveaxis** (*bool*) – Move the last axis to the first dimension before reshaping to a dictionary, e.g. if the batch dimensions are leading.
- **validate** (*bool*) – Validate the input at some cost to performance.

Returns **values** – Mapping from parameter names to values.

Return type dict[str, np.ndarray]

class sciutils.stats.**SemiBoundedVariable** (*loc=0*, *scale=1*)

A semi-bounded variable $y = loc + scale \times \exp(x)$ on the interval $[loc, \inf]$ if $scale > 0$ and $[-\inf, loc]$ if $scale < 0$.

apply (*x*)

Transform a variable from an unconstrained space to a possibly constrained space.

Parameters **x** (*array_like*) – Variable to transform.

Returns

- **y** (*array_like*) – Transformed variable.
- **log_jacobian** (*array_like*) – Logarithm of the Jacobian associated with the transform.

invert (*y*)

Transform a variable from a possibly constrained space to an untransformed space.

Parameters **y** (*array_like*) – Transformed variable.

Returns **x** – Variable after inverse transform.

Return type array_like

sciutils.stats.**cauchy_logcdf** (*x*, *mu*, *sigma*)

Evaluate the log CDF of the Cauchy distribution.

sciutils.stats.**cauchy_logpdf** (*x*, *mu*, *sigma*)

Evaluate the log PDF of the Cauchy distribution.

sciutils.stats.**evaluate_hpd_levels** (*pdf*, *pvals*)

Evaluate the levels that include a given fraction of the the probability mass.

Parameters

- **pdf** (*array_like*) – Probability density function evaluated over a regular grid.
- **pvals** (*array_like or int*) – Probability mass to be included within the corresponding level or the number of levels.

Returns **levels** – Contour levels of the probability density function that enclose the desired probability mass.

Return type array_like

`sciutils.stats.`**`evaluate_hpd_mass`**(*pdf*)

Evaluate the highest posterior density mass excluded from isocontours.

Parameters **pdf** (*array_like*) – Probability density function evaluated over a regular grid.

Returns **excluded** – The probability mass excluded at a given isocontour of the *pdf*.

Return type array_like

`sciutils.stats.`**`evaluate_mode`**(*x*, *lin=200*, ***kwargs*)

Evaluate the mode of a univariate distribution based on samples using a kernel density estimate.

Parameters

- **x** (*array_like*) – Univariate samples from the distribution.
- **lin** (*array_like or int*) – Sample points at which to evaluate the density estimate or the number of sample points across the range of the data.
- ****kwargs** (*dict*) – Additional arguments passed to the `scipy.stats.gaussian_kde` constructor.

Returns **mode**

Return type float

`sciutils.stats.`**`halfcauchy_logcdf`**(*x*, *mu*, *sigma*)

Evaluate the log CDF of the half-Cauchy distribution.

`sciutils.stats.`**`halfcauchy_logpdf`**(*x*, *mu*, *sigma*)

Evaluate the log PDF of the half-Cauchy distribution.

`sciutils.stats.`**`maybe_build_model`**(*model_code*, *root='.pystan'*, ***kwargs*)

Build a pystan model or retrieve a cached version.

Parameters

- **model_code** (*str*) – Stan model code to build.
- **root** (*str*) – Root directory at which to cache models.
- ****kwargs** (*dict*) – Additional arguments passed to the *pystan.StanModel* constructor.

Returns **model** – Compiled stan model.

Return type pystan.StanModel

`sciutils.stats.`**`normal_logcdf`**(*x*, *mu*, *sigma*)

Evaluate the log CDF of the normal distribution.

`sciutils.stats.`**`normal_logpdf`**(*x*, *mu*, *sigma*)

Evaluate the log PDF of the normal distribution.

CHAPTER

THREE

GENERAL UTILITY FUNCTIONS

`sciutils.util.`**`bincountnd`**(*xs*, *weights=None*, *minshape=None*)

Count number of occurrences of each value in an array of non-negative integer tuples.

Parameters

- **xs** (*array_like, 2 dimensions, non-negative integers*) – Input array with shape *(p, n)*, where *p* is the number of dimensions of the output array and *n* is the number of indices.
- **weights** (*array_like, 1 dimension, optional*) – Weight vector of length *n* corresponding to indices in *xs*.
- **minshape** (*tuple, optional*) – A minimum shape for the output array.

Returns **out** – The result of binning the input indices.

Return type ndarray

`sciutils.util.`**`dict_list_transpose`**(*xs*)

Transpose a dictionary of arrays to an array of dictionaries and vice versa.

Parameters **xs** (*mapping or iterable*) – Dictionary of arrays or array of dictionaries to convert.

Returns **y** – Transposed dictionary of arrays or array of dictionaries to convert.

Return type iterable or mapping

PYTHON MODULE INDEX

S

INDEX