spherical_kde Documentation

Release 0.1.0

Will Handley

Oct 18, 2021

CONTENTS:

CHAPTER

ONE

SPHERICAL_KDE

1.1 spherical_kde package

1.1.1 Subpackages

spherical_kde.tests package

Submodules

spherical_kde.tests.test_distributions module

spherical_kde.tests.test_distributions.**random_VonMisesFisher_distribution**()

spherical_kde.tests.test_distributions.**random_phi_theta_sigma**()

spherical_kde.tests.test_distributions.**test_VonMisesFisher_distribution_mean**()

spherical_kde.tests.test_distributions.**test_VonMisesFisher_distribution_normalisation**()

spherical_kde.tests.test_distributions.**test_VonMisesFisher_mean**()

spherical_kde.tests.test_distributions.**test_VonMisesFisher_standarddeviation**()

spherical_kde.tests.test_kde module

spherical_kde.tests.test_kde.**random_kde**(*nsamples*)

spherical_kde.tests.test_kde.**test_kde_bandwith_automatic**()

spherical_kde.tests.test_kde.**test_kde_correct**()

spherical_kde.tests.test_kde.**test_kde_incorrect_lengths**()

spherical_kde.tests.test_kde.**test_kde_incorrect_projection**()

spherical_kde.tests.test_kde.**test_kde_lengths**()

spherical_kde.tests.test_kde.**test_kde_normalised**()

spherical_kde.tests.test_kde.**test_kde_plotting**()

spherical_kde.tests.test_utils module

spherical_kde.tests.test_utils.**test_cartesian_from_polar_array**()

spherical_kde.tests.test_utils.**test_cartesian_from_polar_scalar**()

spherical_kde.tests.test_utils.**test_decra_from_polar_array**()

spherical_kde.tests.test_utils.**test_decra_from_polar_scalar**()

spherical_kde.tests.test_utils.**test_logsinh**()

spherical_kde.tests.test_utils.**test_logsinh_positive_arg**()

spherical_kde.tests.test_utils.**test_polar_from_cartesian_array**()

spherical_kde.tests.test_utils.**test_polar_from_cartesian_scalar**()

spherical_kde.tests.test_utils.**test_polar_from_decra_array**()

spherical_kde.tests.test_utils.**test_polar_from_decra_scalar**()

spherical_kde.tests.test_utils.**test_rotation_matrix**()

spherical_kde.tests.test_utils.**test_spherical_integrate**()

spherical_kde.tests.test_utils.**test_spherical_kullback_liebler**()

Module contents

1.1.2 Submodules

1.1.3 spherical_kde.distributions module

Module containing the kernel function for the spherical KDE.

For more detail, see: https://en.wikipedia.org/wiki/Von_Mises-Fisher_distribution

spherical_kde.distributions.**VonMisesFisher_distribution**(*phi*, *theta*, *phi0*, *theta0*, *sigma0*)
Von-Mises Fisher distribution function.

Parameters

phi, theta [float or array_like] Spherical-polar coordinates to evaluate function at.

phi0, theta0 [float or array-like] Spherical-polar coordinates of the center of the distribution.

sigma0 [float] Width of the distribution.

Returns

float or array_like log-probability of the vonmises fisher distribution.

Notes

Wikipedia: https://en.wikipedia.org/wiki/Von_Mises-Fisher_distribution

spherical_kde.distributions.**VonMisesFisher_sample**(*phi0*, *theta0*, *sigma0*, *size=None*)
Draw a sample from the Von-Mises Fisher distribution.

Parameters

phi0, theta0 [float or array-like] Spherical-polar coordinates of the center of the distribution.

sigma0 [float] Width of the distribution.

size [int, tuple, array-like] number of samples to draw.

Returns

phi, theta [float or array_like] Spherical-polar coordinates of sample from distribution.

spherical_kde.distributions.**VonMises_mean**(*phi*, *theta*)
Von-Mises sample mean.

Parameters

phi, theta [array-like] Spherical-polar coordinate samples to compute mean from.

Returns

float

..math:: sum_i^N x_i / || sum_i^N x_i ||

Notes

Wikipedia: https://en.wikipedia.org/wiki/Von_Mises-Fisher_distribution#Estimation_of_parameters

spherical_kde.distributions.**VonMises_std**(*phi*, *theta*)
Von-Mises sample standard deviation.

Parameters

phi, theta [array-like] Spherical-polar coordinate samples to compute mean from.

Returns

solution for ..math:: 1/tanh(x) - 1/x = R,

where

..math:: R = || sum_i^N x_i || / N

Notes

Wikipedia: https://en.wikipedia.org/wiki/Von_Mises-Fisher_distribution#Estimation_of_parameters but re-parameterised for sigma rather than kappa.

1.1.4 spherical_kde.utils module

Utilities

- General stable functions
- Transforming coordinates
- Computing rotations
- Performing spherical integrals

spherical_kde.utils.**cartesian_from_polar**(*phi*, *theta*)
Embedded 3D unit vector from spherical polar coordinates.

Parameters

phi, theta [float or numpy.array] azimuthal and polar angle in radians.

Returns

nhat [numpy.array] unit vector(s) in direction (phi, theta).

spherical_kde.utils.**decra_from_polar**(*phi*, *theta*)
Convert from ra and dec to spherical polar coordinates.

Parameters

phi, theta [float or numpy.array] azimuthal and polar angle in radians

Returns

ra, dec [float or numpy.array] Right ascension and declination in degrees.

spherical_kde.utils.**logsinh**(*x*)
Compute log(sinh(x)), stably for large x.

Parameters

x [float or numpy.array] argument to evaluate at, must be positive

Returns

float or numpy.array log(sinh(x))

spherical_kde.utils.**polar_from_cartesian**(*x*)
Embedded 3D unit vector from spherical polar coordinates.

Parameters

x [array_like] cartesian coordinates

Returns

phi, theta [float or numpy.array] azimuthal and polar angle in radians.

spherical_kde.utils.**polar_from_decra**(*ra*, *dec*)
Convert from spherical polar coordinates to ra and dec.

Parameters

ra, dec [float or numpy.array] Right ascension and declination in degrees.

Returns

phi, theta [float or numpy.array] Spherical polar coordinates in radians

spherical_kde.utils.**rotation_matrix**(*a*, *b*)
The rotation matrix that takes a onto b.

Parameters

a, b [numpy.array] Three dimensional vectors defining the rotation matrix

Returns

M [numpy.array] Three by three rotation matrix

Notes

StackExchange post: https://math.stackexchange.com/questions/180418/calculate-rotation-matrix-to-align-vector-a-to-vector-b

spherical_kde.utils.**spherical_integrate**(*f*, *log=False*)
Integrate an area density function over the sphere.

Parameters

f [callable] function to integrate (phi, theta) -> float

log [bool] Should the function be exponentiated?

Returns

float Spherical area integral

$$\int_0^{2\pi} d\phi \int_0^{\pi} d\theta f(\phi, \theta) \sin(\theta)$$

spherical_kde.utils.**spherical_kullback_liebler**(*logp*, *logq*)
Compute the spherical Kullback-Liebler divergence.

Parameters

logp, logq [callable] log-probability distributions (phi, theta) -> float

Returns

float Kullback-Liebler divergence

$$\int P(x) \log \frac{P(x)}{Q(x)} dx$$

Notes

Wikipedia post: https://en.wikipedia.org/wiki/Kullback-Leibler_divergence

1.1.5 Module contents

The spherical kernel density estimator class.

class spherical_kde.**SphericalKDE**(*phi_samples*, *theta_samples*, *weights=None*, *bandwidth=None*, *density=100*)

Bases: object

Spherical kernel density estimator

Parameters

phi_samples, theta_samples [array_like] spherical-polar samples to construct the kde

weights [array_like] Sample weighting default [1] * len(phi_samples))

bandwidth [float] bandwidth of the KDE. Increasing bandwidth increases smoothness

density [int] number of grid points in theta and phi to draw contours.

Attributes

phi, theta [numpy.array] spherical polar samples

weights [numpy.array] Sample weighting (normalised to sum to 1).

bandwidth [float] Bandwidth of the kde. defaults to rule-of-thumb estimator https://en.wikipedia.org/wiki/Kernel_density_estimation Set to None to use this value

density [int] number of grid points in theta and phi to draw contours.

palefactor [float] getdist-style colouration factor of sigma-contours.

Methods

__call__(phi, theta)	Log-probability density estimate
plot(ax[, colour])	Plot the KDE on an axis.
plot_samples(ax[, nsamples])	Plot equally weighted samples on an axis.

property bandwidth

plot(*ax*, *colour='g'*, ***kwargs*)

Plot the KDE on an axis.

Parameters

ax [matplotlib.axes.Axes] matplotlib axis to plot on. This must be constructed with a *cartopy.crs.projection*:

```
>>> import cartopy
>>> import matplotlib.pyplot as plt
>>> fig = plt.subplots()
>>> ax = fig.add_subplot(111, projection=cartopy.crs.Mollweide())
```

color Colour to plot the contours. *arg* can be an *RGB* or *RGBA* sequence or a string in any of several forms:

1) a letter from the set 'rgbcmykw'

2) a hex color string, like '#00FFFF'

3) a standard name, like 'aqua'

4) a string representation of a float, like '0.4',

This is passed into *matplotlib.colors.colorConverter.to_rgb*

plot_samples(*ax*, *nsamples=None*, ***kwargs*)

Plot equally weighted samples on an axis.

Parameters

ax [matplotlib.axes.Axes] matplotlib axis to plot on. This must be constructed with a *cartopy.crs.projection*:

```
>>> import cartopy
>>> import matplotlib.pyplot as plt
>>> fig = plt.subplots()
>>> ax = fig.add_subplot(111, projection=cartopy.crs.Mollweide())
```

nsamples [int] Approximate number of samples to plot. Can only thin down to this number, not bulk up

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

V