Package ‘DirStats’

January 10, 2021

Type Package
Title Nonparametric Methods for Directional Data
Version 0.1.7
Date 2021-01-09
License GPL-3
LazyData true
Depends R (>= 3.6.0)
Imports movMF, rotasym
Suggests viridisLite
URL https://github.com/egarpor/DirStats
BugReports https://github.com/egarpor/DirStats
Encoding UTF-8
RoxygenNote 7.1.1
NeedsCompilation yes
Author Eduardo García-Portugués [aut, cre]
Maintainer Eduardo García-Portugués <edgarcia@est-econ.uc3m.es>
Repository CRAN
Date/Publication 2021-01-10 05:10:17 UTC
**R topics documented:**

DirStats-package .......................................................... 2
bic_vmf_mix ............................................................... 2
bw_dir_cv ................................................................. 4
bw_dir_pi ................................................................. 6
conv ................................................................. 8
int ................................................................. 9
kde_dir .......................................................... 10
lebedev .......................................................... 12
vmf .......................................................... 13

Index 14

| DirStats-package | DirStats — Nonparametric Methods for Directional Data |

**Description**


**Author(s)**

Eduardo García-Portugués.

| bic_vmf_mix | Fitting mixtures of von Mises–Fisher distributions |

**Description**

Fitting mixtures of von Mises–Fisher distributions by the Expectation-Maximization algorithm, with determination of the optimal number of mixture components.

**Usage**

bic_vmf_mix(data, M_bound = ceiling(log(nrow(data))), M_neig = 3,
crit = "BIC", iterative = TRUE, plot_it = FALSE, verbose = FALSE,
kappa_max = 250)
bic_vmf_mix

Arguments

- **data**: directional data, a matrix of size \(c(n, q + 1)\).
- **M_bound**: bound for the number of components in the mixtures. If it is not enough, the search for the mixture with minimum \(\text{crit}\) will continue from \(M_{\text{bound}} + 1\) if \(\text{iterative} = \text{TRUE}\). Defaults to \(\text{ceiling}(\log(\text{nrow(data)}))\).
- **M_neig**: number of neighbors explored around the optimal number of mixture components. Defaults to 3.
- **crit**: information criterion employed, either "BIC" (default), "AICc" or "AIC".
- **iterative**: keep exploring higher number of components if the optimum is attained at \(M_{\text{bound}}\)? Defaults to \(\text{TRUE}\).
- **plot_it**: display an informative plot on the optimization’s grid search? Defaults to \(\text{FALSE}\).
- **verbose**: display fitting progress? Defaults to \(\text{FALSE}\).
- **kappa_max**: maximum value of allowed concentrations, to avoid numerical instabilities. Defaults to \(250\).

Details

See Algorithm 3 in García-Portugués (2013). The Expectation-Maximization fit is performed with \texttt{movMF}.

Value

A list with entries:

- **best_fit**: a list with estimated mixture parameters \(\hat{\mu}, \hat{\kappa}\), and \(\hat{p}\) of the best-fitting mixture according to \(\text{crit}\).
- **fit_mixs**: a list with of the fitted mixtures.
- **BICs**: a vector with the BICs (or other information criterion) of the fitted mixtures.

References


Examples

```r
# Sample
q <- 2
n <- 300
set.seed(42)
samp <- rbind(rotasym::r_vMF(n = n / 3, mu = c(rep(0, q), 1), kappa = 5),
              rotasym::r_vMF(n = n / 3, mu = c(rep(0, q), -1), kappa = 5),
              rotasym::r_vMF(n = n / 3, mu = c(1, rep(0, q)), kappa = 5))
```
# Mixture fit
bic_vmf_mix(data = samp, plot_it = TRUE, verbose = TRUE)

---

**bw_dir_cv**

*Cross-validation bandwidth selectors for directional data*

## Description

Likelihood and least squares cross-validation bandwidth selectors for kernel density estimation with directional data.

## Usage

```r
bw_dir_lcv(data, h_grid = exp(seq(log(0.05), log(1.5), l = 100)), L = NULL, 
plot_it = FALSE, optim = TRUE, optim_par = 0.25, optim_lower = 0.06, 
optim_upper = 10)
```

```r
bw_dir_lscv(data, h_grid = exp(seq(log(0.05), log(1.5), l = 100)), 
L = NULL, plot_it = FALSE, optim = TRUE, R_code = FALSE, 
optim_par = 0.25, optim_lower = 0.06, optim_upper = 10)
```

## Arguments

- `data` directional data, a matrix of size $c(n,q + 1)$.
- `h_grid` vector of bandwidths for performing a grid search. Defaults to $\exp(\text{seq}(\log(0.05), \log(1.5), l = 100))$.
- `L` kernel function. Set internally to $\text{function}(x) \exp(-x)$ (von Mises–Fisher kernel) if NULL (default).
- `plot_it` display an informative plot on the optimization’s grid search? Defaults to FALSE.
- `optim` run an optimization? Defaults to TRUE. Otherwise, a grid search on $h$ is done. Only effective if $L = \text{NULL}$.
- `optim_par`, `optim_lower`, `optim_upper` parameters passed to `par`, `lower`, and `upper` in `optim` when using the “L-BFGS-B” method. Default to 0.25, 0.06 (to avoid numerical instabilities), and 10.
- `R_code` use slower R code when $L = \text{NULL}$? Defaults to FALSE.

## Details

data is not checked to have unit norm, so the user must be careful. When $L = \text{NULL}$, faster FORTRAN code is employed.

`bw_dir_lscv` employs Monte Carlo integration for $q > 2$, which results in a random output. Use `set.seed` before to avoid it.
Value

A list with entries:

- \textit{h\_opt}: cross-validation bandwidth.
- \textit{h\_grid}: \textit{h\_grid}, if used (otherwise NULL).
- \textit{CV\_opt}: minimum of the CV loss.
- \textit{CV\_grid}: value of the CV function at \textit{h\_grid}, if used (otherwise NULL).

Source

The function \texttt{bw\_dir\_lscv} employs Netlib’s subroutine \texttt{ribesl} for evaluating the modified Bessel function of the first kind. The subroutine is based on a program by Sookne (1973) and was modified by W. J. Cody and L. Stoltz. An earlier version was published in Cody (1983).

References


Examples

```r
# Sample
n <- 25
t <- 2
set.seed(42)
samp <- rotasym::r_vMF(n = n, mu = c(1, rep(0, t)), kappa = 2)

# bw\_dir\_lcv
bw\_dir\_lcv(data = samp, optim = TRUE)$h\_opt
bw\_dir\_lcv(data = samp, optim = FALSE, plot\_it = TRUE)$h\_opt
bw\_dir\_lcv(data = samp, L = function(x) exp(-x))$h\_opt

# bw\_dir\_lscv
set.seed(42)
bw\_dir\_lscv(data = samp, optim = TRUE)$h\_opt
bw\_dir\_lscv(data = samp, optim = FALSE, plot\_it = TRUE)$h\_opt
bw\_dir\_lscv(data = samp, optim = FALSE, R\_code = TRUE)$h\_opt
bw\_dir\_lscv(data = samp, L = function(x) exp(-x))$h\_opt
```
Plug-in bandwidth selectors for directional data

**Description**

Plug-in bandwidth selectors for kernel density estimation with directional data, including Rule-Of-Thumb (ROT), Asymptotic Mixtures (AMI), and Exact Mixtures (EMI).

**Usage**

```r
bw_dir_rot(data)

bw_dir_ami(data, fit_mix = NULL, L = NULL)

R_Psi_mixvmf(q, mu, kappa, p)

bw_dir_emi(data, fit_mix = NULL, optim = TRUE, h_grid = exp(seq(log(0.05), log(1.5), l = 100)), plot_it = TRUE, optim_par = 0.25, optim_lower = 0.06, optim_upper = 10)
```

**Arguments**

- `data`: directional data, a matrix of size $c(n,q + 1)$.
- `fit_mix`: output from `bic_vmf_mix`. Computed internally if NULL (default).
- `L`: kernel function. Set internally to $\text{function}(x) \exp(-x)$ (von Mises–Fisher kernel) if NULL (default).
- `q`: dimension of $S^q$, $q \geq 1$.
- `mu`, `kappa`, `p`: mixture parameters. $\mu$ is the mean matrix of size $c(\text{length}(p),q + 1)$, $\kappa$ is vector of length($p$) concentration parameters, and $p$ is the vector of mixture proportions.
- `optim`: run an optimization? Defaults to TRUE. Otherwise, a grid search on h is done. Only effective if $L = \text{NULL}$.
- `h_grid`: vector of bandwidths for performing a grid search. Defaults to $\text{exp(seq(log(0.05)),log(1.5), l = 100))}$. 
- `plot_it`: display an informative plot on the optimization’s grid search? Defaults to FALSE.
- `optim_par`, `optim_lower`, `optim_upper`: parameters passed to `par`, `lower`, and `upper` in `optim` when using the "L–BFGS–B" method. Default to $0.25$, $0.06$ (to avoid numerical instabilities), and $10$.

**Details**

See Algorithms 1 (AMI) and 2 (EMI) in García-Portugués (2013). The ROT selector is implemented according to Proposition 2, but without the paper’s typo in equation (6), case $q = 2$, where an incorrect extra $\hat{\kappa}$ appears premultiplying $\left(1 + 4\hat{\kappa}^2\right) \sinh(2\hat{\kappa})$ in the denominator.
bw_dir_ami uses \texttt{R_Psi_mixvmf} for computing the curvature term of a mixture of von Mises-Fisher densities.

\texttt{bw_dir_emi} employs Monte Carlo integration for \( q > 2 \), which results in a random output. Use \texttt{set.seed} before to avoid it.

**Value**

Selected bandwidth for \texttt{bw_dir_rot} and \texttt{bw_dir_ami}. \texttt{bw_dir_emi} returns a list with entries:

- \texttt{h_opt}: cross-validation bandwidth.
- \texttt{h_grid}: \texttt{h_grid}, if used (otherwise NULL).
- \texttt{MISE_opt}: minimum of the MISE loss.
- \texttt{MISE_grid}: value of the MISE function at \texttt{h_grid}, if used (otherwise NULL).

**References**


**Examples**

```r
# Sample
n <- 25
def <- 2
set.seed(42)
samp <- rotasym::r_vMF(n = n, mu = c(1, rep(0, q)), kappa = 2)

# Mixture fit
fit_mix <- bic_vmf_mix(data = samp, plot_it = TRUE)

# ROT
bw_dir_rot(samp)

# AMI
bw_dir_ami(samp)
bw_dir_ami(samp, fit_mix = fit_mix)
bw_dir_ami(samp, fit_mix = fit_mix, L = function(x) exp(-x))

# EMI
bw_dir_emi(samp)
bw_dir_emi(samp, fit_mix = fit_mix, optim = FALSE, plot_it = TRUE)
```
Convenience functions

Description
Normalization of data in $R^{q+1}$ to $S^q$. Transformations between $S^1$ and $[0, 2\pi)$, and between $S^2$ and $[0, 2\pi) \times [0, \pi]$.

Usage

```
norm2(x)
normalize(x)
to_cir(th)
to_rad(x)
to_sph(th, ph)
```

Arguments

- `x` matrix or vector, in $S^1$ for `to_cir`.
- `th` vector of angles in $[0, 2\pi)$.
- `ph` vector of angles in $[0, \pi]$.

Value
Euclidean norm (`norm`) and normalized data (`normalize`). Position in $S^1$ (`to_cir`) or in $[0, 2\pi)$ (`to_rad`). Position in $S^2$ (`to_sph`) or in $[0, 2\pi) \times [0, \pi]$ (`to_rad`).

Examples

```
# Normalization
x <- 1:3
norm2(x)
normalize(x)
x <- rbind(1:3, 3:1)
norm2(x)
normalize(x)

# Circular transformations
th <- 1
x <- c(0, 1)
to_rad(to_cir(th))
to_rad(to_cir(c(th, th + 1)))
to_cir(to_rad(x))
to_cir(to_rad(rbind(x, -x)))
```
# Spherical transformations

```r
th <- 2
ph <- 1
x <- c(0, 1, 0)
to_rad(to_sph(th, ph))
to_rad(to_sph(c(th, th + 1),
            c(ph, ph + 1)))
to_sph(to_rad(x)[, 1], to_rad(x)[, 2])
to_sph(to_rad(rbind(x, -x))[1], to_rad(rbind(x, -x))[2])
```

## Integration routines

### Description

Several quadrature rules for integration of functions on $S^1$, $S^2$, and $S^q$, $q \geq 3$.

### Usage

```r
int_cir(f, N = 500, na.rm = TRUE, f_vect = TRUE, ...)
int_sph(f, na.rm = TRUE, f_vect = TRUE, ...)
int_hypsph(f, q, M = 1e+05, na.rm = TRUE, f_vect = TRUE, ...)
```

### Arguments

- **f**: function to be integrated on $S^q$. Must be vectorized and accept matrix inputs of size c(nx, q + 1).
- **N**: Defaults to 5e2.
- **na.rm**: ignore possible NAs arising from the evaluation of f? Defaults to TRUE.
- **f_vect**: can f be called in a vectorized form, with matrix input? Defaults to TRUE.
- **...**: further arguments passed to f.
- **q**: dimension of $S^q$, $q \geq 1$.
- **M**: number of Monte Carlo replicates. Defaults to 1e5.

### Details

- `int_cir` is an extension of equation (4.1.11) in Press et al. (1997), a periodic trapezoidal rule.
- `int_sph` employs the Lebedev quadrature on $S^2$.
- `int_hypsph` implements a Monte Carlo integration on $S^q$.

### Value

A scalar approximating the integral.
References


Examples

# S^1, trapezoidal rule
f <- function(x) rotasym::d_vMF(x = x, mu = c(0, 1), kappa = 2)
int_cir(f = f)

# S^2, Lebedev rule
f <- function(x) rotasym::d_vMF(x = x, mu = c(0, 0, 1), kappa = 2)
int_sph(f = f)

# S^2, Monte Carlo
f <- function(x) rotasym::d_vMF(x = x, mu = c(0, 0, 1), kappa = 2)
int_hypsph(f = f, q = 2)

kde_dir

Directional kernel density estimator

Description

Kernel density estimation with directional data as in the estimator of Bai et al. (1988).

Usage

kde_dir(x, data, h, L = NULL)
c_h(h, q, L = NULL)
lambda_L(L = NULL, q)
b_L(L = NULL, q)
d_L(L = NULL, q)

Arguments

x    evaluation points, a matrix of size c(nx, q + 1).
data  directional data, a matrix of size c(n, q + 1).
h    bandwidth, a scalar for kde_dir. Can be a vector for c_h.
L    kernel function. Set internally to function(x) exp(-x) (von Mises–Fisher kernel) if NULL (default).
q    dimension of S^q, q ≥ 1.
Details

data is not checked to have unit norm, so the user must be careful. When \( L = \text{NULL} \), faster FORTRAN code is employed.

Value

kde_dir returns a vector of size \( nx \) with the evaluated kernel density estimator. \( c_h \) returns the normalizing constant for the kernel, a vector of length \( \text{length}(h) \). \( \lambda_L \), \( b_L \), and \( d_L \) return moments of \( L \).

References


Examples

```r
# Sample
n <- 50
q <- 3
samp <- rotasym::r_vMF(n = n, mu = c(1, rep(0, q)), kappa = 2)

# Evaluation points
x <- rbind(diag(1, nrow = q + 1), diag(-1, nrow = q + 1))

# kde_dir
kde_dir(x = x, data = samp, h = 0.5, L = NULL)
kde_dir(x = x, data = samp, h = 0.5, L = function(x) exp(-x))

# c_h
c_h(h = 0.5, q = q, L = NULL)
c_h(h = 0.5, q = q, L = function(x) exp(-x))

# b_L
b_L(L = NULL, q = q)
b_L(L = function(x) exp(-x), q = q)

# d_L
d_L(L = NULL, q = q)
d_L(L = function(x) exp(-x), q = q)

# lambda_L
lambda_L(L = NULL, q = q)
lambda_L(L = function(x) exp(-x), q = q)
```
Description

Nodes and weights for Lebedev quadrature on the sphere $S^2$. The rule has 5810 points and is exact up to polynomials of order 131.

Usage

lebedev

Format

A data frame with 5810 rows and two variables:

xyz nodes for quadrature, a matrix with three columns.

w weights for quadrature, a vector.

Details

The approximation to the integral of $f$ has the form

$$\int_{S^2} f(x, y, z) \, dx \, dy \, dz = 4\pi \sum_{i=1}^{N} w_i f(x_i, y_i, z_i)$$

where $N = 5810$. The nodes (in spherical coordinates) and weights are processed from lebedev_131.txt.

Source

https://people.sc.fsu.edu/~jburkardt/datasets/sphere_lebedev_rule/sphere_lebedev_rule.html

References


Examples

```r
# Load data
data("lebedev")

# Integrate x[1] * x[2]^2 (zero integral)
f_1 <- function(x) x[, 1] * x[, 2]^2
4 * pi * sum(lebedev$w * f_1(lebedev$xyz))
```
Von Mises–Fisher distribution utilities

Description
Maximum likelihood estimation for the von Mises–Fisher distribution and evaluation of density mixtures.

Usage

kappa_ml(data)

mu_ml(data)

d_mixvmf(x, mu, kappa, p, norm = FALSE)

Arguments

data
directional data, a matrix of size c(n, q + 1).

x
evaluation points, a matrix of size c(nx, q + 1).

mu, kappa, p
mixture parameters. mu is the mean matrix of size c(length(p), q + 1), kappa is vector of length(p) concentration parameters, and p is the vector of mixture proportions.

norm
enforce normalization of x internally? Defaults to FALSE.

Value
Estimated vector mean (mu_ml) or concentration parameter (kappa_ml). A vector of length nx for d_mixvmf.

Examples

# Sample
n <- 50
q <- 2
samp <- rotasym::r_vMF(n = n, mu = c(1, rep(0, q)), kappa = 2)

# Estimates
mu_ml(samp)
kappa_ml(samp)

# Mixture
x <- to_cir(seq(0, 2 * pi, l = 200))
dens <- d_mixvmf(x = x, mu = rbind(c(-1, 0), c(0, 1), c(1, 0)),
                   kappa = 1:3, p = c(0.5, 0.2, 0.3))
plot(to_rad(x), dens, type = "l")
Index

* datasets
  - lebedev, 12
  - b_L (kde_dir), 10
  - bic_vmf_mix, 2, 6
  - bw_dir_ami (bw_dir_pi), 6
  - bw_dir_cv, 4
  - bw_dir_emi (bw_dir_pi), 6
  - bw_dir_lcv (bw_dir_cv), 4
  - bw_dir_lscv (bw_dir_cv), 4
  - bw_dir_pi, 6
  - bw_dir_rot (bw_dir_pi), 6
  - c_h (kde_dir), 10
  - conv, 8
  - λ_L (kde_dir), 10
  - d_mixvmf (vmf), 13
  - DirStats (DirStats-package), 2
  - DirStats-package, 2
  - int, 9
  - int_cir (int), 9
  - int_hypsph (int), 9
  - int_sph (int), 9
  - kappa_ml (vmf), 13
  - kde_dir, 10
  - lambda_L (kde_dir), 10
  - lebedev, 12
  - Lebedev quadrature, 9
  - movMF, 3
  - mu_ml (vmf), 13
  - norm2 (conv), 8
  - normalize (conv), 8
  - optim, 4, 6
  - R_Psi_mixvmf (bw_dir_pi), 6