Package 'BAMBI'

April 20, 2018

Type Package

Title Bivariate Angular Mixture Models

Version 1.2.1

Date 2018-04-19

Author Saptarshi Chakraborty,
Samuel W.K. Wong

Maintainer Saptarshi Chakraborty <c7rishi@ufl.edu>

Description Fit (using Bayesian methods) and simulate mixtures of univariate and bivariate angular distributions. Chakraborty and Wong (2017) <arXiv:1708.07804> .

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

LinkingTo Rcpp, RcppArmadillo

Imports stats, graphics, Rcpp

Depends R (>= 3.2.0), parallel, label.switching, methods

URL https://arxiv.org/abs/1708.07804

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-20 16:51:25 UTC

R topics documented:

`AIC.angmcmc` *AIC and BIC for angmcmc objects*

Description

AIC and BIC for angmcmc objects

Usage

```
## S3 method for class 'angmcmc'
AIC(object, burnin = 1/3, thin = 1, k = 2, ...)

## S3 method for class 'angmcmc'
BIC(object, burnin = 1/3, thin = 1, ...)
```

Arguments

`object`	an angmcmc object.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.
`k`	numeric, the *penalty* per parameter to be used; the default `k = 2` is the classical AIC.
`...`	optionally more fitted model objects.

Details

Let $\hat{L}$ be the maximum value of the likelihood function for the model, m be the number of estimated parameters in the model and n be the number of data points. Then AIC and BIC are defined as $AIC = -2\log\hat{L} + mk$ and $BIC = -2\log\hat{L} + m\log(n)$.

$\hat{L}$ is estimated by the sample maximum obtained from the MCMC realizations.

Value

AIC computes the AIC and BIC computes BIC for angmcmc objects.

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
AIC(fit.vmsin.20)
BIC(fit.vmsin.20)
```

---

bestmodel *Extracting angmcmc object corresponding to the best fitted model in stepwise fits*

Description

Extracting angmcmc object corresponding to the best fitted model in stepwise fits

Usage

```
bestmodel(step_object)
```

Arguments

`step_object`	stepwise fitted object (output of `fit_stepwise_univariate` or `fit_stepwise_bivariate`).

Value

Returns an angmcmc object corresponding to the the best fitted model in step_object.

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.step.15 <- fit_stepwise_bivariate(tim8, "vmsin", start_ncomp = 3,
                                            max_ncomp = 5, n.iter = 15,
                                            ncores = 1)
fit.vmsin.best.15 <- bestmodel(fit.vmsin.step.15)
fit.vmsin.best.15
```

---

circ_cor *Sample circular correlation coefficients*

Description

Sample circular correlation coefficients

Usage

```
circ_cor(x, type = "js")
```

Arguments

x	two column matrix. NA values are not allowed.
type	type of the circular correlation. Must be one of "fl" or "js" or "kendall". See details.

Details

circ_cor calculates the (sample) circular correlation between the columns of x. Two parametric (the Jammalamadaka-Sarma (1988, equation 2.6) form "js", and the Fisher-Lee (1983, Section 3) form "fl") and two non-parametric (two versions of Kendall's tau) correlation coefficients are considered. The first version of Kendall's tau ("tau1") is based on equation 2.1 in Fisher and Lee (1982), whereas the second version ("tau2") is computed using equations 6.7-6.8 in Zhan et al (2017).

The cost-complexity for "js", "fl", "tau2" and "tau1" are $O(n), O(n^2), O(n^2)$ and $O(n^3)$ respectively, where n denotes the number of rows in x. As such, for large n evaluation of "tau1" will be slow.

References

Fisher, N. I. and Lee, A. J. (1982). Nonparametric measures of angular-angular association. Biometrika, 69(2), 315-321.

Fisher, N. I. and Lee, A. J. (1983). A correlation coefficient for circular data. Biometrika, 70(2):327-332.

Jammalamadaka, S. R. and Sarma, Y. (1988). A correlation coefficient for angular variables. Statistical theory and data analysis II, pages 349-364.

Zhan, X., Ma, T., Liu, S., & Shimizu, K. (2017). On circular correlation for data on the torus. Statistical Papers, 1-21.

Examples

```
# generate data from vmsin model
set.seed(1)
dat <- rvmsin(100, 2,3,-0.8,0,0)

# now calculate circular correlation(s) between the 2 columns of dat
circ_cor(dat, type="js")
circ_cor(dat, type="fl")
circ_cor(dat, type="tau1")
circ_cor(dat, type="tau2")
```

`circ_varcor_model` *Analytic circular variances and correlations for bivariate angular models*

Description

Analytic circular variances and correlations for bivariate angular models

Usage

```
circ_varcor_model(model = "vmsin", kappa1 = 1, kappa2 = 1, kappa3 = 0,
  mu1 = 0, mu2 = 0, nsim = 10000)
```

Arguments

`model`	bivariate angular model. Must be one of `"vmsin"`, `"vmcos"`, or `"wnorm2"`.
`kappa1, kappa2, kappa3`	concentration and covariance parameters. Recycled to the same size. kappa3^2 must be < kappa1*kappa2 in the wnorm2 model (see rwnorm2 for a detailed parameterization of `wnorm2`).
`mu1, mu2`	mean parameters. Ignored as they do not play any role in the analytical formulas.
`nsim`	Monte Carlo sample size. Ignored if all of `kappa1`, `kappa2` and `abs(kappa3)` are < 150 or if model = `"wnorm2"`.

Details

The function computes the analytic circular variances and correlations (both Jammalamadaka-Sarma (JS) and Fisher-Lee (FL) forms) for von Mises sine, von Mises cosine and bivariate wrapped normal distributions.

For `wnorm2`, expressions for the circular variances, JS and FL correlation coefficients can be found in Mardia and Jupp (2009), Jammalamadaka and Sarma (1988) and Fisher and Lee (1983) respectively. For `vmsin` and `vmcos` these expressions are provided in Chakraborty and Wong (2018).

Because the analytic expressions in `vmsin` and `vmcos` models involve infinite sums of product of Bessel functions, if any of `kappa1`, `kappa2` and `kappa3` is larger than or equal to 150, IID Monte Carlo with sample size `nsim` is used to approximate `rho_js` for numerical stability. From `rho_js`, `rho_fl` is computed using Corollary 2.2 in Chakraborty and Wong (2018), which makes cost-complexity for the `rho_fl` evaluation to be of order O(`nsim`) for `vmsin` and `vmcos` models. (In general, `rho_fl` evaluation is of order O(`nsim^2`)).

Value

Returns a list with elements `var1`, `var2` (circular variances for the first and second coordinates), `rho_fl` and `rho_js` (circular correlations). See details.

References

Fisher, N. I. and Lee, A. (1983). A correlation coefficient for circular data. Biometrika, 70(2):327-332.

Jammalamadaka, S. R. and Sarma, Y. (1988). A correlation coefficient for angular variables. Statistical theory and data analysis II, pages 349-364.

Mardia, K. and Jupp, P. (2009). Directional Statistics. Wiley Series in Probability and Statistics. Wiley.

Chakraborty, S. and Wong, S, W.K. (2018). On the circular correlation coefficients for bivariate von Mises distributions on a torus. arXiv e-print.

Examples

```
circ_varcor_model("vmsin", kappa1= 1, kappa2 = 2, kappa3 = 3)

# Monte Carlo approximation
set.seed(1)
dat <- rvmsin(1000, 1, 2, 3)
# sample circular variance
circ_var <- function(x)
  1 - mean(cos(x - atan2(mean(sin(x)), mean(cos(x))) ))
circ_var(dat[, 1])
circ_var(dat[, 2])
circ_cor(dat, "fl")
circ_cor(dat, "js")
```

`contour.angmcmc` *Contour plot for angmcmc objects with bivariate data*

Description

Contour plot for angmcmc objects with bivariate data

Usage

```
## S3 method for class 'angmcmc'
contour(x, fn = mean, show.data = TRUE, nlevels = 20,
  levels, burnin = 1/3, thin = 1, cex = 1, col = "red", ...)
```

Arguments

`x`	angular MCMC object (with bivariate data).
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior mean.
`show.data`	logical. Should the data points be added to the contour plot? Ignored if `object` is NOT supplied.
`nlevels`	number of contour levels desired **if** levels is not supplied; passed to the contour function in graphics.
`levels`	numeric vector of levels at which to draw contour lines; passed to the contour function in graphics.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` = n, only every n-th realizations of the Markov chain is kept.
`cex, col`	graphical parameters passed to `points` in graphics for plotting the data points. Ignored if show.data == FALSE.
`...`	additional arguments to be passed to the function `contour`.

Details

`contour.angmcmc` is an S3 function for angmcmc objects that calls `contour` from graphics.

To estimate the mixture density required to construct the contour plot, first the parameter vector η is estimated by applying `fn` on the MCMC samples, yielding the (consistent) Bayes estimate $\hat{\eta}$. Then the mixture density $f(x|\eta)$ at any point x is (consistently) estimated by $f(x|\hat{\eta})$.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# now create a contour plot with the default first 1/3 as burn-in and thin = 1
contour(fit.vmsin.20)
```

`densityplot1d` *Density curve for angmcmc object with univariate data*

Description

Density curve for angmcmc object with univariate data

Usage

```
densityplot1d(object, fn = mean, show.hist = TRUE, burnin = 1/3,
  thin = 1, ...)
```

Arguments

`object`	angular MCMC object (with univariate data).
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior mean.
`show.hist`	logical. Should a histogram for the data points be added to the plot?
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.
`...`	other arguments to be passed to the function `hist` from graphics. Ignored if `show.hist == FALSE`.

Details

To estimate the mixture density, first the parameter vector η is estimated by applying `fn` on the MCMC samples, yielding the (consistent) Bayes estimate $\hat{\eta}$. Then the mixture density $f(x|\eta)$ at any point x is (consistently) estimated by $f(x|\hat{\eta})$.

Examples

```
# first fit a vm mixture model
# illustration only - more iterations needed for convergence
fit.vm.20 <- fit_vmmix(wind, ncomp = 2, n.iter =  20,
                       ncores = 1)
# now create density curve with the default first 1/3 as burn-in and thin = 1
densityplot1d(fit.vm.20)
```

`densityplot2d` *Density surface for angmcmc objects with bivariate data*

Description

Density surface for angmcmc objects with bivariate data

Usage

```
densityplot2d(object, fn = mean, burnin = 1/3, thin = 1,
  log.density = FALSE, theta = 30, phi = 30, shade = 0.01,
  expand = 0.5, ...)
```

Arguments

`object`	angular MCMC object (with bivariate data).
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior mean.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.
`log.density`	logical. Should log density be used for the plot?
`theta, phi, shade, expand`	arguments passed to `persp` from `graphics`.
`...`	additional arguments passed to `persp` from `graphics`.

Details

`densityplot2d` is a wrapper for `persp` from graphics applied on angmcmc objects.

To estimate the mixture density, first the parameter vector η is estimated by applying `fn` on the MCMC samples, yielding the (consistent) Bayes estimate $\hat{\eta}$. Then the mixture density $f(x|\eta)$ at any point x is (consistently) estimated by $f(x|\hat{\eta})$.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# now create density surface with the default first 1/3 as burn-in and thin = 1
densityplot2d(fit.vmsin.20)
# the viewing angles can be changed through the arguments theta and phi
# (passed to persp from graphics)
densityplot2d(fit.vmsin.20, theta = 45, phi = 45)
```

DIC *Deviance Information Criterion (DIC) for angmcmc objects*

Description

Deviance Information Criterion (DIC) for angmcmc objects

Usage

```
DIC(object, form = 1, burnin = 1/3, thin = 1, ...)
```

Arguments

`object`	angular MCMC object.
`form`	form of DIC to use. Available choices are 1 (default) and 2. See details.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` = n, only every n-th realizations of the Markov chain is kept.
`...`	additional model specific arguments to be passed to `DIC`. For example, `int.displ` specifies integer dispacement in wnorm and wnorm2 models. See fit_wnormmix and fit_wnorm2mix for more details.

Details

Given a deviance function $D(\theta) = -2log(p(y|\theta))$, and an estimate $\theta* = (\sum \theta_i)/N$ of the posterior mean $E(\theta|y)$, where y denote the data, θ are the unknown parameters of the model, $\theta_1, ..., \theta_N$ are MCMC samples from the posterior distribution of θ given y and $p(y|\theta)$ is the likelihood function, the (form 1 of) Deviance Infomation Criterion (DIC) is defined as

$$DIC = 2((\sum_{s=1}^{N} D(\theta_s))/N - D(\theta*))$$

The second form for DIC is given by

$$DIC = D(\theta*) - 4\hat{var} \log p(y|\theta_s)$$

where for $i = 1, ..., n$, $\hat{var} \log p(y|\theta)$ denotes the estimated variance of the log likelihood based on the realizations $\theta_1, ..., \theta_N$.

Like AIC and BIC, DIC is an asymptotic approximation for large samples, and is only valid when the posterior distribution is approximately normal.

Value

Computes the DIC for a given angmcmc object

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
DIC(fit.vmsin.20)
```

`d_fitted` *Density and random deviates from an angmcmc object*

Description

Density and random deviates from an angmcmc object

Usage

```
d_fitted(x, object, fn = mean, burnin = 1/3, thin = 1)

r_fitted(n, object, fn = mean, burnin = 1/3, thin = 1)
```

Arguments

`x`	vector (if univariate) or a two column matrix (if bivariate, with each row a 2-D vector) of points where the densities are to be computed.
`object`	angular MCMC object. The dimension of the model must match with `x`.
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior mean.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.
`n`	number of observations to be generated.

Details

To estimate the mixture density, first the parameter vector η is estimated by applying `fn` on the MCMC samples (using the function pointest), yielding the (consistent) Bayes estimate $\hat{\eta}$. Then the mixture density $f(x|\eta)$ at any point x is (consistently) estimated by $f(x|\hat{\eta})$.

The random deviates are generated from the estimated mixture density $f(x|\hat{\eta})$.

Value

`d_fitted` gives a vector the densities computed at the given points and `r_fitted` creates a vector (if univariate) or a matrix (if bivariate) with each row being a 2-D point, of random deviates.

Examples

```
set.seed(1)
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
d_fitted(c(0,0), fit.vmsin.20)
r_fitted(10, fit.vmsin.20)
```

---

extractsamples *Extract MCMC samples for parameters from an angmcmc object*

Description

Extract MCMC samples for parameters from an angmcmc object

Usage

```
extractsamples(object, par.name = NULL, comp.label = NULL,
               burnin = 1/3, thin = 1)
```

Arguments

`object`	angular MCMC object
`par.name`	vector of names of parameters for which point estimates are to be computed. If `NULL`, results for all parameters are provided.
`comp.label`	vector of component labels (positive integers, e.g., `1, 2, ...`) for which point estimates are to be computed. If `NULL`, results for all components are provided.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.

Details

The default for both `par.name` and `comp.label` are the all possible choices available in `object`.

Value

Returns a matrix (vector) if one (both) of `par.name` and `comp.label` is of length 1, and a three dimensional array with third dimension corresponding to iterations if both `par.name` and `comp.label` have length >1.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# extract Markov chain realizations for kappa1 from component 1
extr_kappa1_1 <- extractsamples(fit.vmsin.20, "kappa1", 1)
# for kappa1 from component from all components
extr_kappa1 <- extractsamples(fit.vmsin.20, "kappa1")
# for all parameters in component 1
extr_1 <- extractsamples(fit.vmsin.20, comp.label = 1)
```

---

fit_stepwise_bivariate

Component size selection via stepwise incremented bivariate mixture models

---

Description

Component size selection via stepwise incremented bivariate mixture models

Usage

```
fit_stepwise_bivariate(data, model, fn = mean,  start_ncomp = 1,
                       max_ncomp = 10, crit = "WAIC", form = 1,
                       prev.par = FALSE, burnin = 1/3, thin = 1, ...)
```

Arguments

`data`	two column matrix (each row being a bivariate vector) of observations. If outside, the values are transformed into the scale $[0, 2\pi)$.
`model`	bivariate angular mixture model to be fitted. One of `"vmsin"`, `"vmcos"`, or `"wnorm2"`.
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior means. Ignored if `prev.par` is `FALSE`.
`start_ncomp`	starting component size. A single component model is fitted if `start_ncomp` is equal to one.
`max_ncomp`	maximum number of components allowed in the mixture model.

crit	criteria for model comparison, one of "AIC", "BIC", "DIC" or "WAIC". Default is "WAIC".
form	form of crit to be used. Available choices are 1 and 2. Used only if crit is "WAIC" or "DIC" and ignored otherwise.
prev.par	logical. Should the final parameters from the model with ncomp = K be used in the model with ncomp = K+1 as the starting parameters?
burnin	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
thin	positive integer. If thin = n, only every n-th realizations of the Markov chain is kept.
...	additional arguments passed to fit_vmsinmix or fit_vmcosmix or fit_wnorm2mix, depending on model.

Details

The default method is "hmc". Each fit uses the number of MCMC iterations specified by the argument n.iter (passed to fit_vmsinmix or fit_vmcosmix or fit_wnorm2mix, as specified through the argument model).

Value

Returns a named list with the following seven elements:

fit.all - a list all angmcmc objects created at each component size;

fit.best - angmcmc object corresponding to the optimum component size;

ncomp.best - optimum component size (integer);

crit - which model comparison criterion used (one of "AIC", "BIC", "DIC" or "WAIC");

crit.all - all crit values calculated (for all component sizes);

crit.best - crit value for the optimum component size; and

check_min - logical; is the optimum component size less than max_ncomp?

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.step.15 <- fit_stepwise_bivariate(tim8, "vmsin", start_ncomp = 3,
                                            max_ncomp = 5, n.iter = 15,
                                            ncores = 1)
(fit.vmsin.best.15 <- bestmodel(fit.vmsin.step.15))
contour(fit.vmsin.best.15)
```

`fit_stepwise_univariate` *Component size selection via stepwise incremented univariate mixture models*

Description

Component size selection via stepwise incremented univariate mixture models

Usage

```
fit_stepwise_univariate(data, model, fn = mean,  start_ncomp = 1,
                        max_ncomp = 10, crit = "WAIC", form = 1,
                        prev.par = FALSE, burnin = 1/3, thin = 1, ...)
```

Arguments

`data`	vector of observations. If outside, the values are transformed into the scale $[0, 2\pi)$.
`model`	univariate angular mixture model to be fitted. Must be one of `"vm"` or `"wnorm"`.
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior means. Ignored if `prev.par` is `FALSE`.
`start_ncomp`	starting component size. A single component model is fitted if `start_ncomp` is equal to one.
`max_ncomp`	maximum number of components allowed in the mixture model.
`crit`	criteria for model comparison, one of `"AIC"`, `"BIC"`, `"DIC"` or `"WAIC"`. Default is `"WAIC"`.
`form`	form of crit to be used. Available choices are 1 and 2. Used only if crit is `"WAIC"` or `"DIC"` and ignored otherwise.
`prev.par`	logical. Should the final parameters from the model with `ncomp = K` be used in the model with `ncomp = K+1` as the starting parameters?
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.
`...`	additional arguments passed to fit_vmmix or fit_wnormmix, depending on `model`.

Details

The default `method` is `"hmc"`. Each fit uses the number of MCMC iterations specified by the argument `n.iter` (passed to fit_vmmix or fit_wnormmix, as specified through the argument `model`).

Value

Returns a named list with the following seven elements:

`fit.all` - a list all angmcmc objects created at each component size;

`fit.best` - angmcmc object corresponding to the optimum component size;

`ncomp.best` - optimum component size (integer);

`crit` - which model comparison criterion used (one of `"AIC"`, `"BIC"`, `"DIC"` or `"WAIC"`);

`crit.all` - all `crit` values calculated (for all component sizes);

`crit.best` - `crit` value for the optimum component size; and

`check_min` - logical; is the optimum component size less than `max_ncomp`?

Examples

```
# illustration only - more iterations needed for convergence
fit.vm.step.15 <- fit_stepwise_univariate(wind, "vm", start_ncomp = 1,
                                          max_ncomp = 3, n.iter = 15,
                                          ncores = 1)
(fit.vm.best.15 <- bestmodel(fit.vm.step.15))
densityplot1d(fit.vm.best.15)
```

---

fit_vmcosmix *Fitting bivariate von Mises cosine model mixtures using MCMC*

Description

Fitting bivariate von Mises cosine model mixtures using MCMC

Usage

```
fit_vmcosmix(data, ncomp, start_par = list(), method = "hmc",
  epsilon = 0.01, L = 10, epsilon.random = TRUE, L.random = FALSE,
  propscale = rep(0.01, 5), n.iter = 500, gam.loc = 0, gam.scale = 1000,
  pmix.alpha = 1/2, norm.var = 1000, autotune = FALSE, iter.tune = 10,
  ncores, show.progress = TRUE)
```

Arguments

`data`	vector of observations. If outside, the values are transformed into the scale $[0, 2\pi)$.
`ncomp`	number of components in the mixture model. If `comp == 1`, a single component model is fitted.
`start_par`	list with elements `pmix` (ignored if `comp == 1`), `kappa1`, `kappa2`, `mu1` and `mu2`, all being vectors of length same as `ncomp`, providing the starting values; with j-th component of each vector corresponding to the j-th component of the mixture distribution. If missing, moment estimators based on random components are used.

`method`	MCMC strategy to be used for the model paramters: `"hmc"` or `"rwmh"`.
`epsilon, L`	tuning parameters for HMC; ignored if `method = "rwmh"`. `epsilon` (step-size) is a quantity in $[0, 1)$ and `L` (leapfrog steps) is a positive integer.
`epsilon.random`	logical. Should a random value from Uniform(0, `epsilon`) be used for `epsilon` at each iteration? Ignored if `method = "rwmh"`.
`L.random`	logical. Should a random value from discrete Uniform(1,..., `L`) be used for `L` at each iteration? Ignored if `method = "rwmh"`.
`propscale`	tuning parameters for RWMH; a vector of size 5 representing the variances for the proposal normal densities for $\kappa_1, \kappa_2, \kappa_3, \mu_1$ and μ_2 respectively. Ignored if `method = "hmc"`.
`n.iter`	number of iterations for the Markov Chain.
`gam.loc, gam.scale`	location and scale (hyper-) parameters for the gamma prior for `kappa1` and `kappa2`. See dgamma. Defaults are `gam.loc = 0, gam.scale = 1000` that makes the prior non-informative.
`pmix.alpha`	concentration parameter(s) for the Dirichlet prior for `pmix`. Must either be a positive real number, or a vector with positive entries and of the same size as `pmix`. The default is 1/2 which corresponds to the Jeffreys prior.
`norm.var`	variance (hyper-) parameter in the normal prior for `kappa3`. (Prior mean is zero). Default is 1000 that makes the prior non-informative.
`autotune`	logical. Should the Markov chain auto-tune the parameters (`epsilon` in HMC and `propscale` in RWMH) on the basis of acceptances in `iter.tune`? (The iterations used to tune the parameters are discarded.) Set to `FALSE` by default.
`iter.tune`	number of initial iterations used to tune the parameters (`epsilon` in HMC and `propscale` in RWMH). Default is 20. Ignored if `autotune == FALSE`.
`ncores`	number of CPU cores to be used for computing the likelihood, the posterior weight matrix for the Gibbs Sampler of mixture proportions and the gradient in HMC in parallel. Default is all of the available cores (obtained via detectCores). Note that parallelization is implemented using OpenMP. This argument is ignored and the computations are done serially if OpenMP is not available.
`show.progress`	logical. Should a progress bar be included?

Details

`fit_vmcosmix` generates MCMC samples of from vmcos mixture model parameters, and returns an angmcmc object as the output, which can be used as an argument for diagnostics and estimation functions.

Default `method` is `"hmc"`.

If the acceptance rate drops below 5% after 100 or more HMC iterations, `epsilon` is automatically lowered, and the Markov chain is restarted at the current parameter values.

Value

returns an angmcmc object.

Examples

```
# illustration only - more iterations needed for convergence
fit.vmcos.10 <- fit_vmcosmix(tim8, ncomp = 3, n.iter =  10,
                             ncores = 1)
fit.vmcos.10
```

`fit_vmmix` *Fitting univariate von Mises mixtures using MCMC*

Description

Fitting univariate von Mises mixtures using MCMC

Usage

```
fit_vmmix(data, ncomp, start_par = list(), method = "hmc", epsilon = 0.07,
  L = 10, epsilon.random = TRUE, L.random = FALSE, propscale = rep(0.01,
  2), n.iter = 500, gam.loc = 0, gam.scale = 1000, pmix.alpha = 1/2,
  autotune = FALSE, iter.tune = 10, ncores, show.progress = TRUE)
```

Arguments

`data`	vector of observations (in radians). If outside, the values are transformed into the scale $[0, 2\pi)$.
`ncomp`	number of components in the mixture model. If `comp == 1`, a single component model is fitted.
`start_par`	list with elements `pmix` (ignored if `comp == 1`), `kappa` and `mu`, all being vectors of length same as `ncomp`, providing the starting values; with j-th component of each vector corresponding to the j-th component of the mixture distribution. If missing, moment estimators based on random components are used.
`method`	MCMC strategy to be used for the model paramters: `"hmc"` or `"rwmh"`.
`epsilon, L`	tuning parameters for HMC; ignored if `method = "rwmh"`. `epsilon` (step-size) is a quantity in $[0, 1)$ and `L` (leapfrog steps) is a positive integer.
`epsilon.random`	logical. Should a random value from Uniform(0, `epsilon`) be used for `epsilon` at each iteration? Ignored if `method = "rwmh"`.
`L.random`	logical. Should a random value from discrete Uniform(1,..., `L`) be used for `L` at each iteration? Ignored if `method = "rwmh"`.
`propscale`	tuning parameters for RWMH; a vector of size 2 representing the variances for the proposal normal densities for κ and μ respectively. Ignored if `method = "hmc"`.
`n.iter`	number of iterations for the Markov Chain.
`gam.loc, gam.scale`	location and scale (hyper-) parameters for the gamma prior for `kappa`. See dgamma. Defaults are `gam.loc = 0, gam.scale = 1000` that makes the prior non-informative.

pmix.alpha	concentration parameter(s) for the Dirichlet prior for pmix. Must either be a positive real number, or a vector with positive entries and of the same size as pmix. The default is 1/2 which corresponds to the Jeffreys prior.
autotune	logical. Should the Markov chain auto-tune the parameters (epsilon in HMC and propscale in RWMH) on the basis of acceptances in iter.tune? (The iterations used to tune the parameters are discarded.) Set to FALSE by default.
iter.tune	number of initial iterations used to tune the parameters (epsilon in HMC and propscale in RWMH). Default is 20. Ignored if autotune == FALSE.
ncores	number of CPU cores to be used for computing the likelihood, the posterior weight matrix for the Gibbs Sampler of mixture proportions and the gradient in HMC in parallel. Default is all of the available cores (obtained via detectCores). Note that parallelization is implemented using OpenMP. This argument is ignored and the computations are done serially if OpenMP is not available.
show.progress	logical. Should a progress bar be included?

Details

fit_vmmix generates MCMC samples of vm mixture model parameters, and returns an angmcmc object as the output, which can be used as an argument for diagnostics and estimation functions.

Default method is "hmc".

If the acceptance rate drops below 5% after 100 or more HMC iterations, epsilon is automatically lowered, and the Markov chain is restarted at the current parameter values.

Value

returns an angular MCMC object.

Examples

```
# illustration only - more iterations needed for convergence
fit.vm.20 <- fit_vmmix(wind, ncomp = 3, n.iter =  20,
                       ncores = 1)
fit.vm.20
```

---

fit_vmsinmix *Fitting bivariate von Mises sine model mixtures using MCMC*

Description

Fitting bivariate von Mises sine model mixtures using MCMC

Usage

```
fit_vmsinmix(data, ncomp, start_par = list(), method = "hmc",
  epsilon = 0.01, L = 10, epsilon.random = TRUE, L.random = FALSE,
  propscale = rep(0.01, 5), n.iter = 500, gam.loc = 0, gam.scale = 1000,
  pmix.alpha = 1/2, norm.var = 1000, autotune = FALSE, iter.tune = 10,
  ncores, show.progress = TRUE)
```

Arguments

`data`	vector of observations. If outside, the values are transformed into the scale $[0, 2\pi)$.
`ncomp`	number of components in the mixture model. If `comp == 1`, a single component model is fitted.
`start_par`	list with elements `pmix` (ignored if `comp == 1`), `kappa1, kappa2, mu1` and `mu2`, all being vectors of length same as `ncomp`, providing the starting values; with j-th component of each vector corresponding to the j-th component of the mixture distribution. If missing, moment estimators based on random components are used.
`method`	MCMC strategy to be used for the model paramters: `"hmc"` or `"rwmh"`.
`epsilon, L`	tuning parameters for HMC; ignored if `method = "rwmh"`. `epsilon` (step-size) is a quantity in $[0, 1)$ and `L` (leapfrog steps) is a positive integer.
`epsilon.random`	logical. Should a random value from Uniform(0, `epsilon`) be used for `epsilon` at each iteration? Ignored if `method = "rwmh"`.
`L.random`	logical. Should a random value from discrete Uniform(1,..., `L`) be used for `L` at each iteration? Ignored if `method = "rwmh"`.
`propscale`	tuning parameters for RWMH; a vector of size 5 representing the variances for the proposal normal densities for $\kappa_1, \kappa_2, \kappa_3, \mu_1$ and μ_2 respectively. Ignored if `method = "hmc"`.
`n.iter`	number of iterations for the Markov Chain.
`gam.loc, gam.scale`	location and scale (hyper-) parameters for the gamma prior for `kappa1` and `kappa2`. See dgamma. Defaults are `gam.loc = 0, gam.scale = 1000` that makes the prior non-informative.
`pmix.alpha`	concentration parameter(s) for the Dirichlet prior for `pmix`. Must either be a positive real number, or a vector with positive entries and of the same size as `pmix`. The default is 1/2 which corresponds to the Jeffreys prior.
`norm.var`	variance (hyper-) parameter in the normal prior for `kappa3`. (Prior mean is zero). Default is 1000 that makes the prior non-informative.
`autotune`	logical. Should the Markov chain auto-tune the parameters (`epsilon` in HMC and `propscale` in RWMH) on the basis of acceptances in `iter.tune`? (The iterations used to tune the parameters are discarded.) Set to `FALSE` by default.
`iter.tune`	number of initial iterations used to tune the parameters (`epsilon` in HMC and `propscale` in RWMH). Default is 20. Ignored if `autotune == FALSE`.
`ncores`	number of CPU cores to be used for computing the likelihood, the posterior weight matrix for the Gibbs Sampler of mixture proportions and the gradient in HMC in parallel. Default is all of the available cores (obtained via detectCores). Note that parallelization is implemented using OpenMP. This argument is ignored and the computations are done serially if OpenMP is not available.
`show.progress`	logical. Should a progress bar be included?

Details

fit_vmsinmix generates MCMC samples of vmsin mixture model parameters, and returns an angmcmc object as the output, which can be used as an argument for diagnostics and estimation functions.

Default method is "hmc".

If the acceptance rate drops below 5% after 100 or more HMC iterations, epsilon is automatically lowered, and the Markov chain is restarted at the current parameter values.

Value

returns an angmcmc object.

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
fit.vmsin.20
```

---

fit_wnorm2mix *Fitting bivariate wrapped normal mixtures using MCMC*

Description

Fitting bivariate wrapped normal mixtures using MCMC

Usage

```
fit_wnorm2mix(data, ncomp, start_par = list(), method = "hmc",
  epsilon = 0.005, L = 10, epsilon.random = TRUE, L.random = FALSE,
  propscale = rep(0.01, 5), n.iter = 500, int.displ, gam.loc = 0,
  gam.scale = 1000, norm.var = 1000, pmix.alpha = 1/2, autotune = FALSE,
  iter.tune = 10, ncores, show.progress = TRUE)
```

Arguments

data	vector of observations. If outside, the values are transformed into the scale $[0, 2\pi)$.
ncomp	number of components in the mixture model. If comp == 1, a single component model is fitted.
start_par	list with elements pmix (ignored if comp == 1), kappa1, kappa2, mu1 and mu2, all being vectors of length same as ncomp, providing the starting values; with j-th component of each vector corresponding to the j-th component of the mixture distribution. If missing, moment estimators based on random components are used.

`method`	MCMC strategy to be used for the model paramters: `"hmc"` or `"rwmh"`.
`epsilon, L`	tuning parameters for HMC; ignored if `method = "rwmh"`. `epsilon` (step-size) is a quantity in $[0, 1)$ and `L` (leapfrog steps) is a positive integer.
`epsilon.random`	logical. Should a random value from Uniform(0, `epsilon`) be used for `epsilon` at each iteration? Ignored if `method = "rwmh"`.
`L.random`	logical. Should a random value from discrete Uniform(1,..., `L`) be used for `L` at each iteration? Ignored if `method = "rwmh"`.
`propscale`	tuning parameters for RWMH; a vector of size 5 representing the variances for the proposal normal densities for $\kappa_1, \kappa_2, \kappa_3, \mu_1$ and μ_2 respectively. Ignored if `method = "hmc"`.
`n.iter`	number of iterations for the Markov Chain.
`int.displ`	absolute integer displacement for each coordinate. Default is 3. Allowed minimum and maximum are 1 and 5 respectively.
`gam.loc, gam.scale`	location and scale (hyper-) parameters for the gamma prior for `kappa1` and `kappa2`. See dgamma. Defaults are `gam.loc = 0, gam.scale = 1000` that makes the prior non-informative.
`norm.var`	variance (hyper-) parameter in the normal prior for `kappa3`. (Prior mean is zero). Default is 1000 that makes the prior non-informative.
`pmix.alpha`	concentration parameter(s) for the Dirichlet prior for `pmix`. Must either be a positive real number, or a vector with positive entries and of the same size as `pmix`. The default is 1/2 which corresponds to the Jeffreys prior.
`autotune`	logical. Should the Markov chain auto-tune the parameters (`epsilon` in HMC and `propscale` in RWMH) on the basis of acceptances in `iter.tune`? (The iterations used to tune the parameters are discarded.) Set to `FALSE` by default.
`iter.tune`	number of initial iterations used to tune the parameters (`epsilon` in HMC and `propscale` in RWMH). Default is 20. Ignored if `autotune == FALSE`.
`ncores`	number of CPU cores to be used for computing the likelihood, the posterior weight matrix for the Gibbs Sampler of mixture proportions and the gradient in HMC in parallel. Default is all of the available cores (obtained via detectCores). Note that parallelization is implemented using OpenMP. This argument is ignored and the computations are done serially if OpenMP is not available.
`show.progress`	logical. Should a progress bar be included?

Details

`fit_wnorm2mix` generates MCMC samples of wnorm2 mixture model parameters, and returns an angmcmc object as the output, which can be used as an argument for diagnostics and estimation functions.

Default `method` is `"hmc"`.

If the acceptance rate drops below 5% after 100 or more HMC iterations, `epsilon` is automatically lowered, and the Markov chain is restarted at the current parameter values.

Value

returns an angmcmc object.

Examples

```
# illustration only - more iterations needed for convergence
fit.wnorm2.15 <- fit_wnorm2mix(tim8, ncomp = 3, n.iter =  15,
                               ncores = 1)
fit.wnorm2.15
```

`fit_wnormmix` *Fitting univariate wrapped normal mixtures using MCMC*

Description

Fitting univariate wrapped normal mixtures using MCMC

Usage

```
fit_wnormmix(data, ncomp, start_par, method = "hmc", epsilon = 0.07,
  L = 10, epsilon.random = TRUE, L.random = FALSE, propscale = rep(0.01,
  2), n.iter = 10000, int.displ, gam.loc = 0, gam.scale = 1000,
  pmix.alpha = 1/2, autotune = FALSE, iter.tune = 10, ncores,
  show.progress = TRUE)
```

Arguments

`data`	vector of observations (in radians). If outside, the values are transformed into the scale $[0, 2\pi)$.
`ncomp`	number of components in the mixture model. If `comp == 1`, a single component model is fitted.
`start_par`	list with elements `pmix` (ignored if `comp == 1`), `kappa` and `mu`, all being vectors of length same as `ncomp`, providing the starting values; with j-th component of each vector corresponding to the j-th component of the mixture distribution. If missing, moment estimators based on random components are used.
`method`	MCMC strategy to be used for the model paramters: `"hmc"` or `"rwmh"`.
`epsilon, L`	tuning parameters for HMC; ignored if `method = "rwmh"`. `epsilon` (step-size) is a quantity in $[0, 1)$ and `L` (leapfrog steps) is a positive integer.
`epsilon.random`	logical. Should a random value from Uniform(0, `epsilon`) be used for `epsilon` at each iteration? Ignored if `method = "rwmh"`.
`L.random`	logical. Should a random value from discrete Uniform(1,..., `L`) be used for `L` at each iteration? Ignored if `method = "rwmh"`.
`propscale`	tuning parameters for RWMH; a vector of size 2 representing the variances for the proposal normal densities for κ and μ respectively. Ignored if `method = "hmc"`.

`n.iter` number of iterations for the Markov Chain.

`int.displ` integer displacement. The allowed values are 1, 2, 3, 4 and 5. Default is 3.

`gam.loc, gam.scale` location and scale (hyper-) parameters for the gamma prior for `kappa`. See dgamma. Defaults are `gam.loc = 0, gam.scale = 1000` that makes the prior non-informative.

`pmix.alpha` concentration parameter(s) for the Dirichlet prior for `pmix`. Must either be a positive real number, or a vector with positive entries and of the same size as `pmix`. The default is 1/2 which corresponds to the Jeffreys prior.

`autotune` logical. Should the Markov chain auto-tune the parameters (`epsilon` in HMC and `propscale` in RWMH) on the basis of acceptances in `iter.tune`? (The iterations used to tune the parameters are discarded.) Set to `FALSE` by default.

`iter.tune` number of initial iterations used to tune the parameters (`epsilon` in HMC and `propscale` in RWMH). Default is 20. Ignored if `autotune == FALSE`.

`ncores` number of CPU cores to be used for computing the likelihood, the posterior weight matrix for the Gibbs Sampler of mixture proportions and the gradient in HMC in parallel. Default is all of the available cores (obtained via detectCores). Note that parallelization is implemented using OpenMP. This argument is ignored and the computations are done serially if OpenMP is not available.

`show.progress` logical. Should a progress bar be included?

Details

`fit_wnormmix` generates MCMC samples of wnorm mixture model parameters, and returns an angmcmc object as the output, which can be used as an argument for diagnostics and estimation functions.

Default `method` is `"hmc"`.

If the acceptance rate drops below 5% after 100 or more HMC iterations, `epsilon` is automatically lowered, and the Markov chain is restarted at the current parameter values.

Value

returns an angular MCMC object.

Examples

```
# illustration only - more iterations needed for convergence
fit.wnorm.20 <- fit_wnormmix(wind, ncomp = 3, n.iter =  20,
                             ncores = 1)
fit.wnorm.20
```

`fix_label` *Fix label switching in angmcmc objects*

Description

Fix label switching in angmcmc objects

Usage

```
fix_label(object, burnin = 1/3, thin = 1, method = 1)
```

Arguments

`object`	angular MCMC object.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept.
`method`	method to use for fixing label switching. Available choices are 1 for dataBased and 2 for ecr.iterative.1.

Details

`fix_label` is a wrapper for dataBased or ecr.iterative.1 (depending on `method`) from the `label.switching` package for angmcmc objects.

Value

Returns another angmcmc object, with the parameter values (after burn-in and thin) re-ordered according to the resulting permutation from dataBased or ecr.iterative.1.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# now apply fix_label
fit.vmsin.20.fix <- fix_label(fit.vmsin.20)
```

is.angmcmc *Angular MCMC (*angmcmc*) Object*

Description

Checking if an R object is angmcmc

Usage

```
is.angmcmc(object)
```

Arguments

object	any R object

Details

angmcmc objects are classified lists that are created when any of the five mixture model fitting functions, viz., fit_vmmix, fit_wnormmix, fit_vmsinmix, fit_vmcosmix and fit_wnorm2mix is used. An angmcmc object contains a number of elements, including the dataset, the model being fitted on the dataset and dimension of the model (univariate or bivariate), the tuning parameters used, MCMC samples for the mixture model parameters, the (hidden) component or cluster indicators for data points in each iteration and the (iteration-wise) log likelihood and log posterior density values (both calculated upto some normalizing constants). When printed, an angmcmc object returns a brief summary of the function arguments used to produce the object and the average acceptance rate of the proposals (in HMC and RWMH) used over iterations. An angmcmc object can be used as an argument for the diagnostic and post-processing functions available in BAMBI for making further inferences.

Value

logical. Is the input an angmcmc object?

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
is.angmcmc(fit.vmsin.20)
```

`logLik.angmcmc` *Extract Log-Likelihood from angmcmc objects*

Description

Extract Log-Likelihood from angmcmc objects

Usage

```
## S3 method for class 'angmcmc'
logLik(object, fn = mean, burnin = 1/3, thin = 1, ...)
```

Arguments

`object`	angular MCMC object.
`fn`	function to evaluate on MCMC samples to estimate parameters. Defaults to `mean`, which computes the estimated posterior mean. Used for parameter estimation.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1). Used for parameter estimation.
`thin`	positive integer. If `thin` $= n$, only every n-th realizations of the Markov chain is kept. Used for parameter estimation.
`...`	some methods for this generic function require additional arguments.

Details

In order to estimate the log likelihood for the model, first the parameter vector is estimated using pointest, and then log the likelihood is calculated on the basis of the estimated parameter.

The degrees of the likelihood function is the total number of free parameters estimated in the mixture models, which is equal to $6K - 1$ for bivariate models (vmsin, vmcos and wnorm2), or $3K - 1$ for univariate models (vm and wnorm), where K denotes the number of components in the mixture model.

Value

Returns an object of class logLik. This is a number (the estimated log likelihood) with attributes "df" (degrees of freedom) and "nobs" (number of observations).

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
logLik(fit.vmsin.20)
```

lpdtrace *Trace plot of log posterior density or log likelihood from an angmcmc object*

Description

Trace plot of log posterior density or log likelihood from an angmcmc object

Usage

```
lpdtrace(object, use.llik = FALSE, burnin = 1/3, thin = 1)
```

Arguments

object	angular MCMC object.
use.llik	logical. Should log likelihood be plotted instead of log posterior? Set to FALSE by default.
burnin	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
thin	positive integer. If thin = n, only every n-th realizations of the Markov chain is kept.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# log posterior density trace
lpdtrace(fit.vmsin.20)
# log likelihood trace
lpdtrace(fit.vmsin.20, use.llik = TRUE)
```

paramtrace *Trace plot for parameters from an angmcmc object*

Description

Trace plot for parameters from an angmcmc object

Usage

```
paramtrace(object, par, comp.label, burnin = 1/3, thin = 1)
```

Arguments

object	angular MCMC object.
par	parameter for which trace plot is to be created.
comp.label	vector of component labels (positive integers, e.g., 1, 2, ...) for which point estimates are to be computed. If NULL, results for all components are provided.
burnin	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
thin	positive integer. If thin = n, only every n-th realizations of the Markov chain is kept.

Value

Returns a single plot if a single par and a single comp.label is supplied. Otherwise, a series of plots is produced.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# trace plot for kappa1 in component 1
paramtrace(fit.vmsin.20, "kappa1", 1)
# for kappa1 in all components
paramtrace(fit.vmsin.20, "kappa1")
# for all parameters in component 1
paramtrace(fit.vmsin.20, comp.label = 1)
```

---

pointest *Point estimates for parameters from an angmcmc object*

Description

Point estimates for parameters from an angmcmc object

Usage

```
pointest(object, fn = mean, par.name = NULL, comp.label = NULL,
  burnin = 1/3, thin = 1)
```

Arguments

object	angular MCMC object.
fn	function to evaluate on MCMC samples to estimate parameters. Defaults to mean, which computes the estimated posterior mean.
par.name	vector of names of parameters for which point estimates are to be computed. If NULL, results for all parameters are provided.
comp.label	vector of component labels (positive integers, e.g., 1, 2, ...) for which point estimates are to be computed. If NULL, results for all components are provided.
burnin	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
thin	positive integer. If thin = n, only every n-th realizations of the Markov chain is kept.

Value

Returns a matrix of point estimates, or vector of point estimates if length(par.name)==1 or length(comp.label)==1.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# estimate parameters by sample mean
(est_mean <- pointest(fit.vmsin.20))
# estimate parameters by sample median
(est_median <- pointest(fit.vmsin.20, fn = median))
```

---

quantile.angmcmc *Quantile estimates for parameters from an angmcmc object*

Description

Quantile estimates for parameters from an angmcmc object

Usage

```
## S3 method for class 'angmcmc'
quantile(x, par.name = NULL, comp.label = NULL,
  burnin = 1/3, thin = 1, ...)
```

Arguments

x	angmcmc object
par.name	vector of names of parameters for which point estimates are to be computed. If NULL, results for all parameters are provided.
comp.label	vector of component labels (positive integers, e.g., 1, 2, ...) for which point estimates are to be computed. If NULL, results for all components are provided.
burnin	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
thin	positive integer. If thin = n, only every n-th realizations of the Markov chain is kept.
...	further arguments to pass to quantile. In particular, probs = seq(0, 1, 0.25) is the default vector of quantiles computed for each parameter.

Value

Returns a three dimensional array of quantiles, or a matrix (vector) of quantiles if one (or two) among par.name, comp.label, probs has length 1.

Examples

```
# first fit a vmsin mixture model
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
# 0.025th quantiles
(quant_025 <- quantile(fit.vmsin.20, prob = 0.025))
# 0.975th quantiles
(quant_975 <- quantile(fit.vmsin.20, prob = 0.975))
# default quantiles
(quant_def <- quantile(fit.vmsin.20))
```

---

rvm *The univariate von Mises distribution*

Description

The univariate von Mises distribution

Usage

```
rvm(n, kappa = 1, mu = 0)

dvm(x, kappa = 1, mu = 0)
```

Arguments

n	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
kappa	vector of concentration (inverse-variance) parameters; kappa > 0.
mu	vector of means.
x	vector of angles (in radians) where the densities are to be evaluated.

Details

If mu and kappa are not specified they assume the default values of 0 and 1 respectively.

The univariate von Mises distribution has density

$$f(x) = 1/(2\pi I_0(\kappa)) \exp(\kappa \cos(x - mu))$$

where $I_0(\kappa)$ denotes the modified Bessel function of the first kind with order 0 evaluated at the point κ.

Value

dvm gives the density and rvm generates random deviates.

Examples

```
kappa <- 1:3
mu <- 0:2
x <- 1:10
n <- 10


# when x and both parameters are scalars, dvm returns a single density
dvm(x[1], kappa[1], mu[1])

# when x is a vector but both the parameters are scalars, dmv returns a vector of
# densities calculated at each entry of x with the same parameters
dvm(x, kappa[1], mu[1])

# if x is scalar and at least one of the two paraemters is a vector, both parameters are
# recycled to the same length, and dvm returns a vector of with ith element being the
# density evaluated at x with parameter values kappa[i] and mu[i]
dvm(x[1], kappa, mu)

# if x and at least one of the two paraemters is a vector, x and the two parameters are
# recycled to the same length, and dvm returns a vector of with ith element being the
# density at ith element of the (recycled) x with parameter values kappa[i] and mu[i]
dvm(x, kappa, mu)

# when parameters are all scalars, number of observations generated by rvm is n
rvm(n, kappa[1], mu[1])
```

```
# when at least one of the two parameters is a vector, both are recycled to the same length,
# n is ignored, and the number of observations generated by rvm is the same as the length of
# the recycled vectors
rvm(n, kappa, mu)
```

`rvmcos` *The bivariate von Mises cosine model*

Description

The bivariate von Mises cosine model

Usage

```
rvmcos(n, kappa1 = 1, kappa2 = 1, kappa3 = 0, mu1 = 0, mu2 = 0)

dvmcos(x, kappa1 = 1, kappa2 = 1, kappa3 = 0, mu1 = 0, mu2 = 0)
```

Arguments

`n`	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
`kappa1, kappa2, kappa3`	vectors of concentration parameters; `kappa1, kappa2 > 0`.
`mu1, mu2`	vectors of mean parameters.
`x`	bivariate vector or a two-column matrix with each row being a bivariate vector of angles (in radians) where the densities are to be evaluated.

Details

The bivariate von Mises cosine model density at the point $x = (x_1, x_2)$ is given by

$$f(x) = C_c(\kappa_1, \kappa_2, \kappa_3) \exp(\kappa_1 \cos(T_1) + \kappa_2 \cos(T_2) + \kappa_3 \cos(T_1 - T_2))$$

where

$$T_1 = x_1 - \mu_1; T_2 = x_2 - \mu_2$$

and $C_c(\kappa_1, \kappa_2, \kappa_3)$ denotes the normalizing constant for the cosine model.

Value

`dvmcos` gives the density and `rvmcos` generates random deviates.

Examples

```
kappa1 <- c(1, 2, 3)
kappa2 <- c(1, 6, 5)
kappa3 <- c(0, 1, 2)
mu1 <- c(1, 2, 5)
mu2 <- c(0, 1, 3)
x <- diag(2, 2)
n <- 10

# when x is a bivariate vector and parameters are all scalars,
# dvmcos returns single density
dvmcos(x[1, ], kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# when x is a two column matrix and parameters are all scalars,
# dmvsin returns a vector of densities calculated at the rows of
# x with the same parameters
dvmcos(x, kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# if x is a bivariate vector and at least one of the parameters is
# a vector, all parameters are recycled to the same length, and
# dvmcos returns a vector with ith element being the density
# evaluated at x with parameter values kappa1[i], kappa2[i],
# kappa3[i], mu1[i] and mu2[i]
dvmcos(x[1, ], kappa1, kappa2, kappa3, mu1, mu2)

# if x is a two column matrix and at least one of the parameters is
# a vector, rows of x and the parameters are recycled to the same
# length, and dvmcos returns a vector with ith element being the
# density evaluated at ith row of x with parameter values kappa1[i],
# kappa2[i], # kappa3[i], mu1[i] and mu2[i]
dvmcos(x[1, ], kappa1, kappa2, kappa3, mu1, mu2)

# when parameters are all scalars, number of observations generated
# by rvmcos is n
rvmcos(n, kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# when at least one of the parameters is a vector, all parameters are
# recycled to the same length, n is ignored, and the number of
# observations generated by rvmcos is the same as the length of the
# recycled vectors
rvmcos(n, kappa1, kappa2, kappa3, mu1, mu2)
```

`rvmcosmix` *The bivariate von Mises cosine model mixtures*

Description

The bivariate von Mises cosine model mixtures

Usage

```
rvmcosmix(n, kappa1, kappa2, kappa3, mu1, mu2, pmix)

dvmcosmix(x, kappa1, kappa2, kappa3, mu1, mu2, pmix)
```

Arguments

`n`	number of observations.
`kappa1, kappa2, kappa3`	vectors of concentration parameters; `kappa1, kappa2 > 0` for each component.
`mu1, mu2`	vectors of mean parameters.
`pmix`	vector of mixture proportions.
`x`	matrix of angles (in radians) where the density is to be evaluated, with each row being a single bivariate vector of angles.

Details

All the argument vectors `pmix, kappa1, kappa2, kappa3, mu1` and `mu2` must be of the same length (= component size of the mixture model), with j-th element corresponding to the j-th component of the mixture distribution.

The bivariate von Mises cosine model mixture distribution with component size `K = length(pmix)` has density

$$g(x) = \sum p[j] * f(x; \kappa_1[j], \kappa_2[j], \kappa_3[j], \mu_1[j], \mu_2[j])$$

where the sum extends over j; $p[j]$; $\kappa_1[j], \kappa_2[j], \kappa_3[j]$; and $\mu_1[j], \mu_2[j]$ respectively denote the mixing proportion, the three concentration parameters and the two mean parameter for the j-th cluster, $j = 1, ..., K$, and $f(.; \kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2)$ denotes the density function of the von Mises cosine model with concentration parameters $\kappa_1, \kappa_2, \kappa_3$ and mean parameters μ_1, μ_2.

Value

`dvmcosmix` computes the density and `rvmcosmix` generates random deviates from the mixture density.

Examples

```
kappa1 <- c(1, 2, 3)
kappa2 <- c(1, 6, 5)
kappa3 <- c(0, 1, 2)
mu1 <- c(1, 2, 5)
mu2 <- c(0, 1, 3)
pmix <- c(0.3, 0.4, 0.3)
x <- diag(2, 2)
n <- 10

# mixture densities calculated at the rows of x
dvmcosmix(x, kappa1, kappa2, kappa3, mu1, mu2, pmix)

# number of observations generated from the mixture distribution is n
```

```
rvmcosmix(n, kappa1, kappa2, kappa3, mu1, mu2, pmix)
```

`rvmmix` *The univariate von Mises mixtures*

Description

The univariate von Mises mixtures

Usage

```
rvmmix(n, kappa, mu, pmix)

dvmmix(x, kappa, mu, pmix)
```

Arguments

`n`	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
`kappa`	vector of component concentration (inverse-variance) parameters, `kappa > 0`.
`mu`	vector of component means.
`pmix`	vector of mixing proportions.
`x`	vector of angles (in radians) where the densities are to be evaluated.

Details

`pmix`, `mu` and `kappa` must be of the same length, with j-th element corresponding to the j-th component of the mixture distribution.

The univariate von Mises mixture distribution with component size `K = length(pmix)` has density

$$g(x) = p[1] * f(x; \kappa[1], \mu[1]) + ... + p[K] * f(x; \kappa[K], \mu[K])$$

where $p[j], \kappa[j], \mu[j]$ respectively denote the mixing proportion, concentration parameter and the mean parameter for the j-th component and $f(.; \kappa, \mu)$ denotes the density function of the (univariate) von Mises distribution with mean parameter μ and concentration parameter κ.

Value

`dvmmix` computes the density and `rvmmix` generates random deviates from the mixture density.

Examples

```
kappa <- 1:3
mu <- 0:2
pmix <- c(0.3, 0.3, 0.4)
x <- 1:10
n <- 10

# mixture densities calculated at each point in x
dvmmix(x, kappa, mu, pmix)

# number of observations generated from the mixture distribution is n
rvmmix(n, kappa, mu, pmix)
```

`rvmsin` *The bivariate von Mises sine model*

Description

The bivariate von Mises sine model

Usage

```
rvmsin(n, kappa1 = 1, kappa2 = 1, kappa3 = 0, mu1 = 0, mu2 = 0)

dvmsin(x, kappa1 = 1, kappa2 = 1, kappa3 = 0, mu1 = 0, mu2 = 0)
```

Arguments

`n`	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
`kappa1, kappa2, kappa3`	vectors of concentration parameters; `kappa1, kappa2 > 0`.
`mu1, mu2`	vectors of mean parameters.
`x`	bivariate vector or a two-column matrix with each row being a bivariate vector of angles (in radians) where the densities are to be evaluated.

Details

The bivariate von Mises sine model density at the point $x = (x_1, x_2)$ is given by

$$f(x) = C_s(\kappa_1, \kappa_2, \kappa_3) \exp(\kappa_1 \cos(T_1) + \kappa_2 \cos(T_2) + \kappa_3 \sin(T_1) \sin(T_2))$$

where

$$T_1 = x_1 - \mu_1; T_2 = x_2 - \mu_2$$

and $C_s(\kappa_1, \kappa_2, \kappa_3)$ denotes the normalizing constant for the sine model.

Value

dvmsin gives the density and rvmsin generates random deviates.

Examples

```
kappa1 <- c(1, 2, 3)
kappa2 <- c(1, 6, 5)
kappa3 <- c(0, 1, 2)
mu1 <- c(1, 2, 5)
mu2 <- c(0, 1, 3)
x <- diag(2, 2)
n <- 10

# when x is a bivariate vector and parameters are all scalars,
# dvmsin returns single density
dvmsin(x[1, ], kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# when x is a two column matrix and parameters are all scalars,
# dmvsin returns a vector of densities calculated at the rows of
# x with the same parameters
dvmsin(x, kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# if x is a bivariate vector and at least one of the parameters is
# a vector, all parameters are recycled to the same length, and
# dvmsin returns a vector of with ith element being the density
# evaluated at x with parameter values kappa1[i], kappa2[i],
# kappa3[i], mu1[i] and mu2[i]
dvmsin(x[1, ], kappa1, kappa2, kappa3, mu1, mu2)

# if x is a two column matrix and at least one of the parameters is
# a vector, rows of x and the parameters are recycled to the same
# length, and dvmsin returns a vector of with ith element being the
# density evaluated at ith row of x with parameter values kappa1[i],
# kappa2[i], # kappa3[i], mu1[i] and mu2[i]
dvmsin(x[1, ], kappa1, kappa2, kappa3, mu1, mu2)

# when parameters are all scalars, number of observations generated
# by rvmsin is n
rvmsin(n, kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# when at least one of the parameters is a vector, all parameters are
# recycled to the same length, n is ignored, and the number of
# observations generated by rvmsin is the same as the length of the
# recycled vectors
rvmsin(n, kappa1, kappa2, kappa3, mu1, mu2)
```

---

rvmsinmix *The bivariate von Mises sine model mixtures*

Description

The bivariate von Mises sine model mixtures

Usage

```
rvmsinmix(n, kappa1, kappa2, kappa3, mu1, mu2, pmix)

dvmsinmix(x, kappa1, kappa2, kappa3, mu1, mu2, pmix)
```

Arguments

n	number of observations.
kappa1, kappa2, kappa3	vectors of concentration parameters; kappa1, kappa2 > 0 for each component.
mu1, mu2	vectors of mean parameters.
pmix	vector of mixture proportions.
x	matrix of angles (in radians) where the density is to be evaluated, with each row being a single bivariate vector of angles.

Details

All the argument vectors pmix, kappa1, kappa2, kappa3, mu1 and mu2 must be of the same length (= component size of the mixture model), with j-th element corresponding to the j-th component of the mixture distribution.

The bivariate von Mises sine model mixture distribution with component size K = length(p.mix) has density

$$g(x) = \sum p[j] * f(x; \kappa_1[j], \kappa_2[j], \kappa_3[j], \mu_1[j], \mu_2[j])$$

where the sum extends over j; $p[j]$; $\kappa_1[j], \kappa_2[j], \kappa_3[j]$; and $\mu_1[j], \mu_2[j]$ respectively denote the mixing proportion, the three concentration parameters and the two mean parameter for the j-th component, $j = 1, ..., K$, and $f(.; \kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2)$ denotes the density function of the von Mises sine model with concentration parameters $\kappa_1, \kappa_2, \kappa_3$ and mean parameters μ_1, μ_2.

Value

dvmsinmix computes the density (vector if x is a two column matrix with more than one row) and rvmsinmix generates random deviates from the mixture density.

Examples

```
kappa1 <- c(1, 2, 3)
kappa2 <- c(1, 6, 5)
kappa3 <- c(0, 1, 2)
mu1 <- c(1, 2, 5)
mu2 <- c(0, 1, 3)
pmix <- c(0.3, 0.4, 0.3)
x <- diag(2, 2)
n <- 10
```

```
# mixture densities calculated at the rows of x
dvmsinmix(x, kappa1, kappa2, kappa3, mu1, mu2, pmix)

# number of observations generated from the mixture distribution is n
rvmsinmix(n, kappa1, kappa2, kappa3, mu1, mu2, pmix)
```

`rwnorm` *The univariate Wrapped Normal distribution*

Description

The univariate Wrapped Normal distribution

Usage

```
rwnorm(n = 1, kappa = 1, mu = 0)

dwnorm(x, kappa = 1, mu = 0, int.displ)
```

Arguments

`n`	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
`kappa`	vector of concentration (inverse-variance) parameters; `kappa` > 0.
`mu`	vector of means.
`x`	vector of angles (in radians) where the densities are to be evaluated.
`int.displ`	integer displacement. The allowed values are 1, 2, 3, 4 and 5. Default is 3.

Details

If `mu` and `kappa` are not specified they assume the default values of `0` and `1` respectively.

The univariate wrapped normal distribution has density

$$f(x) = \sqrt{(}\kappa/(2\pi)) \sum \exp(-\kappa/2(x - \mu(2\pi\omega))^2)$$

where the sum extends over all integers ω, and is approximated by a sum over ω in $\{-M, -M + 1, ..., M - 1, M\}$ if `int.displ =` M.

Value

`dwnorm` gives the density and `rwnorm` generates random deviates.

Examples

```
kappa <- 1:3
mu <- 0:2
x <- 1:10
n <- 10


# when x and both parameters are scalars, dwnorm returns a single density
dwnorm(x[1], kappa[1], mu[1])

# when x is a vector but both the parameters are scalars, dmv returns a vector of
# densities calculated at each entry of x with the same parameters
dwnorm(x, kappa[1], mu[1])

# if x is scalar and at least one of the two paraemters is a vector, both parameters are
# recycled to the same length, and dwnorm returns a vector of with ith element being the
# density evaluated at x with parameter values kappa[i] and mu[i]
dwnorm(x[1], kappa, mu)

# if x and at least one of the two paraemters is a vector, x and the two parameters are
# recycled to the same length, and dwnorm returns a vector of with ith element being the
# density at ith element of the (recycled) x with parameter values kappa[i] and mu[i]
dwnorm(x, kappa, mu)

# when parameters are all scalars, number of observations generated by rwnorm is n
rwnorm(n, kappa[1], mu[1])

# when at least one of the two parameters is a vector, both are recycled to the same length,
# n is ignored, and the number of observations generated by rwnorm is the same as the length
# of the recycled vectors
rwnorm(n, kappa, mu)
```

`rwnorm2` *The bivariate Wrapped Normal distribution*

Description

The bivariate Wrapped Normal distribution

Usage

```
rwnorm2(n, kappa1 = 1, kappa2 = 1, kappa3 = 0, mu1 = 0, mu2 = 0)

dwnorm2(x, kappa1 = 1, kappa2 = 1, kappa3 = 0, mu1 = 0, mu2 = 0,
  int.displ)
```

Arguments

n	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
kappa1, kappa2, kappa3	vectors of concentration parameters; kappa1, kappa2 > 0, and kappa3^2 < kappa1*kappa2.
mu1, mu2	vectors of mean parameters.
x	bivariate vector or a two-column matrix with each row being a bivariate vector of angles (in radians) where the densities are to be evaluated.
int.displ	integer displacement. The allowed values are 1, 2, 3, 4 and 5. Default is 3.

Details

The bivariate wrapped normal density at the point $x = (x_1, x_2)$ is given by,

$$f(x) = \sqrt{((\kappa_1\kappa_2 - (\kappa_3)^2))}/(2\pi) \sum \exp(-1/2 * (\kappa_1(T_1)^2 + \kappa_2(T_2)^2 + 2\kappa_3(T_1)(T_2)))$$

where

$$T_1 = T_1(x, \mu, \omega) = (x_1 - \mu_1(2\pi\omega_1))$$
$$T_2 = T_2(x, \mu, \omega) = (x_2 - \mu_1(2\pi\omega_2))$$

the sum extends over all pairs of integers $\omega = (\omega_1, \omega_2)$, and is approximated by a sum over (ω_1, ω_2) in $\{-M, -M+1, ..., M-1, M\}^2$ if int.displ = M.

Note that above density is essentially the "wrapped" version of a bivariate normal density with mean

$$\mu = (\mu_1, \mu_2)$$

and dispersion matrix $\Sigma = \Delta^{-1}$, where

$$\Delta = \begin{matrix} \kappa_1 & \kappa_3 \\ \kappa_3 & \kappa_2. \end{matrix}$$

Value

dwnorm2 gives the density and rwnorm2 generates random deviates.

Examples

```
kappa1 <- c(1, 2, 3)
kappa2 <- c(1, 6, 5)
kappa3 <- c(0, 1, 2)
mu1 <- c(1, 2, 5)
mu2 <- c(0, 1, 3)
x <- diag(2, 2)
n <- 10

# when x is a bivariate vector and parameters are all scalars,
```

```
# dwnorm2 returns single density
dwnorm2(x[1, ], kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# when x is a two column matrix and parameters are all scalars,
# dmvsin returns a vector of densities calculated at the rows of
# x with the same parameters
dwnorm2(x, kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# if x is a bivariate vector and at least one of the parameters is
# a vector, all parameters are recycled to the same length, and
# dwnorm2 returns a vector of with ith element being the density
# evaluated at x with parameter values kappa1[i], kappa2[i],
# kappa3[i], mu1[i] and mu2[i]
dwnorm2(x[1, ], kappa1, kappa2, kappa3, mu1, mu2)

# if x is a two column matrix and at least one of the parameters is
# a vector, rows of x and the parameters are recycled to the same
# length, and dwnorm2 returns a vector of with ith element being the
# density evaluated at ith row of x with parameter values kappa1[i],
# kappa2[i], # kappa3[i], mu1[i] and mu2[i]
dwnorm2(x[1, ], kappa1, kappa2, kappa3, mu1, mu2)

# when parameters are all scalars, number of observations generated
# by rwnorm2 is n
rwnorm2(n, kappa1[1], kappa2[1], kappa3[1], mu1[1], mu2[1])

# when at least one of the parameters is a vector, all parameters are
# recycled to the same length, n is ignored, and the number of
# observations generated by rwnorm2 is the same as the length of the
# recycled vectors
rwnorm2(n, kappa1, kappa2, kappa3, mu1, mu2)
```

`rwnorm2mix` *The bivariate Wrapped Normal mixtures*

Description

The bivariate Wrapped Normal mixtures

Usage

```
rwnorm2mix(n, kappa1, kappa2, kappa3, mu1, mu2, pmix)

dwnorm2mix(x, kappa1, kappa2, kappa3, mu1, mu2, pmix, int.displ)
```

Arguments

`n`	number of observations.

`kappa1, kappa2, kappa3`
: vectors of concentration parameters; `kappa1, kappa2 > 0, kappa3^2 < kappa1*kappa2` for each component.

`mu1, mu2`
: vectors of mean parameters.

`pmix`
: vector of mixture proportions.

`x`
: matrix of angles (in radians) where the density is to be evaluated, with each row being a single bivariate vector of angles.

`int.displ`
: integer displacement. The allowed values are 1, 2, 3, 4 and 5. Default is 3.

Details

All the argument vectors `pmix, kappa1, kappa2, kappa3, mu1` and `mu2` must be of the same length, with j-th element corresponding to the j-th component of the mixture distribution.

The bivariate wrapped normal mixture distribution with component size `K = length(pmix)` has density

$$g(x) = \sum p[j] * f(x; \kappa_1[j], \kappa_2[j], \kappa_3[j], \mu_1[j], \mu_2[j])$$

where the sum extends over j; $p[j]$; $\kappa_1[j], \kappa_2[j], \kappa_3[j]$; and $\mu_1[j], \mu_2[j]$ respectively denote the mixing proportion, the three concentration parameters and the two mean parameter for the j-th component, $j = 1, ..., K$, and $f(.; \kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2)$ denotes the density function of the wrapped normal distribution with concentration parameters $\kappa_1, \kappa_2, \kappa_3$ and mean parameters μ_1, μ_2.

Value

`dwnorm2mix` computes the density and `rwnorm2mix` generates random deviates from the mixture density.

Examples

```
kappa1 <- c(1, 2, 3)
kappa2 <- c(1, 6, 5)
kappa3 <- c(0, 1, 2)
mu1 <- c(1, 2, 5)
mu2 <- c(0, 1, 3)
pmix <- c(0.3, 0.4, 0.3)
x <- diag(2, 2)
n <- 10

# mixture densities calculated at the rows of x
dwnorm2mix(x, kappa1, kappa2, kappa3, mu1, mu2, pmix)

# number of observations generated from the mixture distribution is n
rwnorm2mix(n, kappa1, kappa2, kappa3, mu1, mu2, pmix)
```

`rwnormmix` *The univariate Wrapped Normal mixtures*

Description

The univariate Wrapped Normal mixtures

Usage

```
rwnormmix(n = 1, kappa, mu, pmix)

dwnormmix(x, kappa, mu, pmix, int.displ)
```

Arguments

`n`	number of observations. Ignored if at least one of the other parameters have length k > 1, in which case, all the parameters are recycled to length k to produce k random variates.
`kappa`	vector of component concentration (inverse-variance) parameters, `kappa > 0`.
`mu`	vector of component means.
`pmix`	vector of mixing proportions.
`x`	vector of angles (in radians) where the densities are to be evaluated.
`int.displ`	integer displacement. The allowed values are 1, 2, 3, 4 and 5. Default is 3.

Details

`pmix`, `mu` and `kappa` must be of the same length, with j-th element corresponding to the j-th component of the mixture distribution.

The univariate wrapped normal mixture distribution with component size `K = length(pmix)` has density

$$g(x) = p[1] * f(x; \kappa[1], \mu[1]) + ... + p[K] * f(x; \kappa[K], \mu[K])$$

where $p[j], \kappa[j], \mu[j]$ respectively denote the mixing proportion, concentration parameter and the mean parameter for the j-th component and $f(.; \kappa, \mu)$ denotes the density function of the (univariate) wrapped normal distribution with mean parameter μ and concentration parameter κ.

Value

`dwnormmix` computes the density and `rwnormmix` generates random deviates from the mixture density.

Examples

```
kappa <- 1:3
mu <- 0:2
pmix <- c(0.3, 0.3, 0.4)
x <- 1:10
n <- 10

# mixture densities calculated at each point in x
dwnormmix(x, kappa, mu, pmix)

# number of observations generated from the mixture distribution is n
rwnormmix(n, kappa, mu, pmix)
```

`summary.angmcmc` *Summary statistics for parameters from an angmcmc object*

Description

Summary statistics for parameters from an angmcmc object

Usage

```
## S3 method for class 'angmcmc'
summary(object, burnin = 1/3, thin = 1, ...)
```

Arguments

`object`	angular MCMC object.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` = n, only every n-th realizations of the Markov chain is kept.
`...`	additional arguments affecting the summary produced.

Details

Computes (after thinning and discarding burn-in) point estimates with 95% posterior credible sets for all components and all parameters, together with the sample averages of log likelihood and log posterior density.

Value

Returns a list with elements `estimate, lower, upper, llik` and `lpd`.

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
summary(fit.vmsin.20)
```

`tim8` *Backbone Dihedral Angles of Triose Phosphate Isomerase (8TIM)*

Description

A dataset consisting of 490 pairs of backbone dihedral angles (in radian scale $[0, 2\pi)$) (ϕ, ψ) for the protein Triose Phosphate Isomerase (8TIM). The angles were obtained first by using the DSSP software on the PDB file for 8TIM to get the backbone angles (in degrees), and then by converting all angles into radians.

Usage

```
data(tim8)
```

Format

A data frame with 490 rows and 2 variables (backbone dihedral angles) phi and psi.

Source

8TIM PDB file: http://www.rcsb.org/pdb/explore.do?structureId=8tim.

DSSP software: http://swift.cmbi.ru.nl/gv/dssp/.

`WAIC` *Watanabe-Akaike Information Criterion (WAIC) for angmcmc objects*

Description

Watanabe-Akaike Information Criterion (WAIC) for angmcmc objects

Usage

```
WAIC(object, form = 1, burnin = 1/3, thin = 1, ...)
```

Arguments

`object`	angular MCMC object.
`form`	the form of p_W to use. Available choices are 1 (default) and 2. See details.
`burnin`	initial fraction of the MCMC samples to be discarded as burn-in. Must be a value in [0, 1).
`thin`	positive integer. If `thin` = n, only every n-th realizations of the Markov chain is kept.
`...`	additional model specific arguments to be passed to `WAIC`. For example, `int.displ` specifies integer dispacement in wnorm and wnorm2 models. See fit_wnormmix and fit_wnorm2mix for more details.

Details

Given a deviance function $D(\eta) = -2\log(p(y|\eta))$, and an estimate $\eta* = (\sum \eta_i)/n$ of the posterior mean $E(\eta|y)$, where $y = (y_1, ..., y_n)$ denote the data, η is the unknown parameter vector of the model, $\eta_1, ..., \eta_N$ are MCMC samples from the posterior distribution of η given y and $p(y|\eta)$ is the likelihood function, the Watanabe-Akaike Information Criterion (WAIC) is defined as

$$WAIC = LPPD - p_W$$

where

$$LPPD = \sum_{i=1}^{n} \log(N^{-1} \sum_{s=1}^{N} p(y_i|\eta_s))$$

and (form 1 of)

$$p_W = 2\sum_{i=1}^{n} [\log(N^{-1} \sum_{s=1}^{N} p(y_i|\eta_s)) - N^{-1} \sum_{s=1}^{N} \log\ p(y_i|\eta_s)].$$

An alternative form (form 2) for p_W is given by

$$p_W = \sum_{i=1}^{n} \hat{var} \log p(y_i|\eta)$$

where for $i = 1, ..., n$, $\hat{var} \log p(y_i|\eta)$ denotes the estimated variance of $\log p(y_i|\eta)$ based on the realizations $\eta_1, ..., \eta_N$.

Value

Computes the WAIC for a given angmcmc object.

Examples

```
# illustration only - more iterations needed for convergence
fit.vmsin.20 <- fit_vmsinmix(tim8, ncomp = 3, n.iter =  20,
                             ncores = 1)
WAIC(fit.vmsin.20)
```

`wind` *Saturna Island wind directions*

Description

A dataset consisting of 239 observations on wind direction in radians (original measurements were in 10s of degrees), measured at Saturna Island, British Columbia, Canada during October 1-10, 2016 (obtained from Environment Canada website). There was a severe storm during October 4-7, which caused significant fluctuations among the wind directions.

Usage

```
data(wind)
```

Format

An object of class `numeric` of length 239.

Source

Environment Canada: http://climate.weather.gc.ca/climate_data/data_quality_e.html.

CBC news on the storm: http://www.cbc.ca/news/canada/british-columbia/storm-bc-1.3795204.

Index