Package 'BayesMultiMode'

August 8, 2023

Type Package

Title Bayesian Mode Inference

Version 0.6.0

Description
A Bayesian approach for mode inference which works in two steps. First, a mixture distribution is fitted on the data using a sparse finite mixture (SFM) Markov chain Monte Carlo (MCMC) algorithm following Malsiner-Walli, Frühwirth-Schnatter and Grün (2016) <doi:10.1007/s11222-014-9500-2>). The number of mixture components does not have to be known; the size of the mixture is estimated endogenously through the SFM approach. Second, the modes of the estimated mixture at each MCMC draw are retrieved using algorithms specifically tailored for mode detection. These estimates are then used to construct posterior probabilities for the number of modes, their locations and uncertainties, providing a powerful tool for mode inference.

License GPL (>= 3)

Imports assertthat, bayesplot, dplyr, ggplot2, ggpubr, gtools, magrittr, MCMCglmm, mvtnorm, posterior, sn, stringr, tidyr, Rdpack, scales

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0)

RdMacros Rdpack

Encoding UTF-8

LazyData true

URL https://github.com/paullabonne/BayesMultiMode

BugReports https://github.com/paullabonne/BayesMultiMode/issues

NeedsCompilation no

RoxygenNote 7.2.3

Config/testthat/edition 3

Author Nalan Baştürk [aut],
Jamie Cross [aut],
Peter de Knijff [aut],

Lennart Hoogerheide [aut],
Paul Labonne [aut, cre],
Herman van Dijk [aut]

Maintainer Paul Labonne <paul.labonne@bi.no>

Repository CRAN

Date/Publication 2023-08-08 10:50:06 UTC

R topics documented:

`bayes_estimation` *Bayesian estimation of mixture distributions*

Description

Gibbs samplers for sparse finite mixture Markov chain Monte Carlo (SFM MCMC) estimation.

Usage

```
bayes_estimation(
  data,
  K,
  dist,
  priors = list(),
  nb_iter = 2000,
  burnin = nb_iter/2,
  printing = TRUE
)
```

Arguments

data	Vector of observations.
K	Maximum number of mixture components.
dist	String indicating the distribution of the mixture components; Currently supports "normal", "skew_normal", "poisson" and "shifted_poisson".
priors	List of priors; default is an empty list which implies the following priors: a0 = 1, A0 = 200, b0 = median(y), B0 = (max(y) - min(y))^2 (normal), D_xi = 1, D_psi =1, (skew normal: B0 = diag(D_xi,D_psi)), c0 = 2.5, l0 = 1.1 (poisson), l0 = 5 (shifted poisson), L0 = 1.1/median(y), L0 = l0 - 1 (shifted poisson), g0 = 0.5, G0 = 100*g0/c0/B0 (normal), G0 = g0/(0.5*var(y)) (skew normal).
nb_iter	Number of MCMC iterations; default is 2000.
burnin	Number of MCMC iterations used as burnin; default is nb_iter/2.
printing	Showing MCMC progression ?

Details

Let y_i, $i = 1, \ldots, n$ denote observations. A general mixture of K distributions from the same parametric family is given by:

$$y_i \sim \sum_{k=1}^{K} \pi_k p(\cdot|\theta_k)$$

with $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$, $k = 1, ..., K$.

The exact number of components does not have to be known a priori when using the SFM MCMC approach. Rather, an upper bound is specified for the number of components and the weights of superfluous components are shrunk towards zero during the estimation. Following Malsiner-Walli et al. (2016) a symmetric Dirichlet prior is used for the mixture weights:

$$\pi_k \sim \text{Dirichlet}(e_0, \ldots, e_0)$$

where a Gamma hyperprior is used on the concentration parameter e_0:

$$e_0 \sim \text{Gamma}\,(a_0, A_0)$$

Mixture of Normal distributions

Normal components take the form:

$$p(y_i|\mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{1}{2}\left(\frac{y_i - \mu_k}{\sigma_k}\right)^2\right)$$

Independent conjugate priors are used for μ_k and σ_k^2 (see for instance Malsiner-Walli et al. 2016) :

$$\mu_k \sim \text{Normal}(\text{b}_0, \text{B}_0),$$

$$\sigma_k^{-2} \sim \text{Gamma}(\text{c}_0, \text{C}_0),$$

$$C_0 \sim \text{Gamma}(\text{g}_0, \text{G}_0).$$

Mixture of skew-Normal distributions

We use the skew-Normal of Azzalini (1985) which takes the form:

$$p(y_i|\xi_k, \omega_k, \alpha_k) = \frac{1}{\omega_k\sqrt{2\pi}}\, \exp\left(-\frac{1}{2}\left(\frac{y_i - \xi_k}{\omega_k}\right)^2\right)\, \left(1 + \text{erf}\left(\alpha_k\left(\frac{y_i - \xi_k}{\omega_k\sqrt{2}}\right)\right)\right)$$

where ξ_k is a location parameter, ω_k a scale parameter and α_k the shape parameter introducing skewness. For Bayesian estimation, we adopt the approach of Fruhwirth-Schnatter and Pyne (2010) and use the following reparameterised random-effect model:

$$z_i \sim TN_{[0,\infty)}(0, 1)$$

$$y_i|(S_i = k) = \xi_k + \psi_k z_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma_k^2)$$

where the parameters of the skew-Normal are recovered with

$$\omega_k = \frac{\psi_k}{\sigma_k}, \qquad \omega_k^2 = \sigma_k^2 + \psi_k^2$$

By defining a regressor $x_i = (1, z_i)'$, the skew-Normal mixture can be seen as random effect model and sampled using standard techniques. Thus we use priors similar to the Normal mixture model:

$$(\xi_k, \psi_k)' \sim \text{Normal}(\text{b}_0, \text{B}_0),$$

$$\sigma_k^{-2} \sim \text{Gamma}(\text{c}_0, \text{C}_0),$$

$$\text{C}_0 \sim \text{Gamma}(\text{g}_0, \text{G}_0).$$

We set

$$\text{b}_0 = (\text{median}(y), 0)'$$

and

$$\text{B}_0 = \text{diag}(\text{D\_xi}, \text{D\_psi})$$

with D_xi = D_psi = 1.

Mixture of Poisson distributions

Poisson components take the form:

$$p(y_i|\lambda_k) = \frac{1}{y_i!}\, \lambda_k^{y_i}\, \exp(-\lambda_k).$$

The prior for λ_k follows from Viallefont et al. (2002):

$$\lambda_k \sim \text{Gamma}(\text{l}_0, \text{L}_0).$$

Mixture of shifted-Poisson distributions

Shifted-Poisson components take the form

$$p(y_i|\lambda_k, \kappa_k) = \frac{1}{(y_i - \kappa_k)!}\, \lambda_k^{(y_i-\kappa_k)!}\, \exp(-\lambda_k)$$

where κ_k is a location or shift parameter with uniform prior.

Value

A list of class `BayesMixture` containing

- data - Same as argument.
- dist_type - Type of the distribution (continuous or discrete).
- pars_names - Names of the mixture components' parameters.
- mcmc - Matrix of MCMC draws where the rows corresponding to burnin have been discarded.
- mcmc_all - Original matrix of MCMC draws.

References

Malsiner-Walli G, Fruhwirth-Schnatter S, Grun B (2016). "Model-based clustering based on sparse finite Gaussian mixtures." *Statistics and Computing*, **26**(1), 303–324. ISSN 1573-1375, doi:10.1007/s1122201495002.

Frühwirth-Schnatter S, Pyne S (2010). "Bayesian inference for finite mixtures of univariate and multivariate skew-normal and skew-t distributions." *Biostatistics*, **11**(2), 317–336. ISSN 1465-4644, doi:10.1093/biostatistics/kxp062.

Frühwirth-Schnatter S, Malsiner-Walli G (2019). "From here to infinity: sparse finite versus Dirichlet process mixtures in model-based clustering." *Advances in Data Analysis and Classification*, **13**, 33–64.

Azzalini A (1985). "A Class of Distributions Which Includes the Normal Ones." *Scandinavian Journal of Statistics*, **12**(2), 171–178. ISSN 0303-6898, Publisher: [Board of the Foundation of the Scandinavian Journal of Statistics, Wiley].

Viallefont V, Richardson S, Peter J (2002). "Bayesian analysis of Poisson mixtures." *Journal of Nonparametric Statistics*, **14**(1-2), 181–202.

Examples

```
# Example with galaxy data ================================================
set.seed(123)

# retrieve galaxy data
```

```
y = galaxy

# estimation
bayesmix = bayes_estimation(data = y,
                            K = 5, #not many to run the example rapidly
                            dist = "normal",
                            nb_iter = 500, #not many to run the example rapidly
                            burnin = 100)

# plot estimated mixture
# plot(bayesmix, max_size = 200)

# Changing priors ================================================
set.seed(123)

# retrieve galaxy data
y = galaxy

# estimation
K = 5
bayesmix = bayes_estimation(data = y,
                            K = K, #not many to run the example rapidly
                            dist = "normal",
                            priors = list(a0 = 10,
                                          A0 = 10*K),
                            nb_iter = 500, #not many to run the example rapidly
                            burnin = 100)

# plot estimated mixture
# plot(bayesmix, max_size = 200)

# Example with DNA data ==========================================================

set.seed(123)

# retrieve DNA data
y = d4z4

# estimation
bayesmix = bayes_estimation(data = y,
                            K = 5, #not many to run the example rapidly
                            dist = "shifted_poisson",
                            nb_iter = 500, #not many to run the example rapidly
                            burnin = 100)

# plot estimated mixture
# plot(bayesmix, max_size = 200)
```

---

bayes_mode *Bayesian mode inference*

Description

This function estimates modes for each mcmc draw and uses these estimates to compute posterior probabilities for the number of modes and their locations (following the approach of Cross et al. 2023). The fixed-point algorithm of Carreira-Perpinan (2000) is used for Gaussian mixtures. The Modal EM algorithm of Li et al. (2007) is used for other continuous mixtures. A basic algorithm is used for discrete mixtures (see Cross et al. 2023).

Usage

```
bayes_mode(
  BayesMix,
  rd = 1,
  tol_x = sd(BayesMix$data)/10,
  tol_conv = 1e-08,
  show_plot = FALSE,
  nb_iter = NULL
)
```

Arguments

`BayesMix`	An object of class `BayesMixture`.
`rd`	Rounding parameter.
`tol_x`	Tolerance parameter for distance in-between modes; default is sd(data)/10 where data is an element of argument `BayesMix`. If two modes are closer than `tol_x`, only the first estimated mode is kept.
`tol_conv`	Tolerance parameter for convergence of the algorithm; default is 1e-8. Not needed for mixtures of discrete distributions.
`show_plot`	Show density with estimated mode as vertical bars ?
`nb_iter`	Number of draws on which the mode-finding algorithm is run; default is NULL which means the algorithm is run on all draws.

Details

Each draw from the MCMC output after burnin, $\theta^{(d)}, \quad d = 1, ..., D$, leads to a posterior predictive probability density/mass function:

$$p(y|\theta^{(d)}) = \sum_{k=1}^{K} \pi_k^{(d)} p(y|\theta_k^{(d)}).$$

Using this function, the mode in draw d $y_m^{(d)}$, $m = 1, ..., M^{(d)}$, where $M^{(d)}$ is the number of modes, are estimated using the algorithm mentioned in the description above.

After running this procedure across all retained posterior draws, we compute the posterior probability for the number of modes being M as:

$$P(\#\text{modes} = M) = \frac{1}{D} \sum_{d=1}^{D} 1(M^{(d)} = M).$$

Similarly, posterior probabilities for locations of the modes are given by:

$$P(y = \text{mode}) = \frac{1}{D} \sum_{d=1}^{D} \sum_{m=1}^{M^{(d)}} 1(y = y_m^{(d)}),$$

for each location y in the range $[\min(y), \max(y)]$. Obviously, continuous data are not defined on a discrete support; it is therefore necessary to choose a rounding decimal to discretize their support (with the `rd` argument).

Value

A list of class `BayesMode` containing

- data - from `BayesMix`.
- dist - from `BayesMix`.
- dist_type - from `BayesMix`.
- pars_names - from `BayesMix`.
- modes - Matrix with a row for each draw and columns showing modes.
- p1 - Posterior probability of unimodality.
- tb_nb_modes - Matrix showing posterior probabilities for the number of modes.
- table_location - Matrix showing the posterior probabilities for location points being modes.

References

Cross JL, Hoogerheide L, Labonne P, van Dijk HK (2023). "Credible Mode Determination in Multimodal Economic and Financial Data Distributions." *Tinbergen Institute Discussion Paper TI 2023-038/III*. Carreira-Perpinan MA (2000). "Mode-finding for mixtures of Gaussian distributions." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(11), 1318–1323. ISSN 1939-3539, doi:10.1109/34.888716, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Li J, Ray S, Lindsay BG (2007). "A Nonparametric Statistical Approach to Clustering via Mode Identification." *Journal of Machine Learning Research*, **8**, 1687-1723.

Examples

```
# Example with galaxy data ================================================
set.seed(123)

# retrieve galaxy data
y = galaxy

# estimation
bayesmix = bayes_estimation(data = y,
                            K = 5, #not many to run the example rapidly
                            dist = "normal",
                            nb_iter = 500, #not many to run the example rapidly
                            burnin = 100)
```

```
# mode estimation
bayesmode = bayes_mode(bayesmix)

# plot
# plot(bayesmode, max_size = 200)

# summary
# summary(bayesmode)

# Example with DNA data ==========================================
set.seed(123)

# retrieve galaxy data
y = d4z4

# estimation
bayesmix = bayes_estimation(data = y,
                            K = 5, #not many to run the example rapidly
                            dist = "shifted_poisson",
                            nb_iter = 500, #not many to run the example rapidly
                            burnin = 100)

# mode estimation
bayesmode = bayes_mode(bayesmix)

# plot
# plot(bayesmode, max_size = 200)

# summary
# summary(bayesmode)

# Example with a Student t ======================================
mu = c(0.5,6)
sigma = c(1,2)
nu = c(5,5)
p = c(0.8,0.2)
params = c(eta = p, mu = mu, sigma = sigma, nu = nu)
pars_names = c("eta", "mu", "sigma", "nu")
dist_type = "continuous"

data = c(sn::rst(p[1]*1000, mu[1], sigma[1], nu = nu[1]),
         sn::rst(p[2]*1000, mu[2], sigma[2], nu = nu[2]))

fit = c(eta = p, mu = mu, sigma = sigma, nu = nu)
fit = rbind(fit, fit)

pdf_func = function(x, pars) {
  sn::dst(x, pars["mu"], pars["sigma"], pars["xi"], pars["nu"])
}

bayesmix = new_BayesMixture(fit, data, K = 2, burnin = 1,
pars_names = pars_names, pdf_func = pdf_func, dist_type = dist_type)
```

```
bayesmode = bayes_mode(bayesmix)

# plot
# plot(bayesmode, max_size = 200)

# summary
# summary(bayesmode)
```

`bayes_trace` *Trace plots*

Description

This is wrapper around the [bayesplot::mcmc_trace()] function from package **bayesplot**.

Usage

```
bayes_trace(BayesMix, mcmc_vars = NULL, with_burnin = FALSE, ...)
```

Arguments

`BayesMix`	An object of class `BayesMixture`.
`mcmc_vars`	Variables to plot; default is all the variable in the MCMC output.
`with_burnin`	Plot all draws ?
`...`	Additional arguments passed to function [bayesplot::mcmc_trace()].

Value

A trace plot.

Examples

```
# Example with galaxy data ================================================
set.seed(123)

# retrieve galaxy data
y = galaxy

# estimation
bayesmix = bayes_estimation(data = y,
                            K = 5, #not many to run the example rapidly
                            dist = "normal",
                            nb_iter = 500, #not many to run the example rapidly
                            burnin = 100)

# trace plot
bayes_trace(bayesmix)
```

ct47 *X chromosomal macrosatellite repeats ct47*

Description

Repeat units that encode for a cancer testis antigen.
Locus (hg18): Xq24
Unit (kb): 4.8
Restriction enzyme: EcoRI
Encoded product : cancer testis antigen 47

Usage

```
ct47
```

Format

A vector of counts with 410 elements.

References

Schaap M, Lemmers RJ, Maassen R, van der Vliet PJ, Hoogerheide LF, van Dijk HK, Basturk N, de Knijff P, van der Maarel SM (2013). "Genome-wide analysis of macrosatellite repeat copy number variation in worldwide populations: evidence for differences and commonalities in size distributions and size restrictions." *BMC Genomics*, **14**(1), 143. ISSN 1471-2164, doi:10.1186/1471216414143.

cyclone *Tropical cyclones lifetime maximum intensity*

Description

Dataset constructed using the International Best Track Archive for Climate Stewardship (IBTrACS). The distribution of tropical cyclones lifetime maximum intensity across the globe is known to be bimodal which has important implications for climate modelling.

Usage

```
cyclone
```

Format

A dataset with three columns showing the identification of the cyclone, its year of occurrence and its lifetime maximum intensity (LMI). LMI is calculated as the maximum wind speed for each cyclone with unit ks.

Source

https://www.ncei.noaa.gov/products/international-best-track-archive

References

Knapp KR, Kruk MC, Levinson DH, Diamond HJ, Neumann CJ (2010). "The International Best Track Archive for Climate Stewardship (IBTrACS): Unifying Tropical Cyclone Data." *Bulletin of the American Meteorological Society*, **91**(3), 363–376. ISSN 0003-0007, 1520-0477, doi:10.1175/2009BAMS2755.1, Publisher: American Meteorological Society Section: Bulletin of the American Meteorological Society.

Knapp KR, Diamond HJ, J.P. K, Kruk MC, Schreck CJ (2018). "International Best Track Archive for Climate Stewardship (IBTrACS) Project, Version 4." *NOAA National Centers for Environmental Information*. doi:10.1175/2009BAMS2755.1.

`d4z4` *Autosomal macrosatellite repeats d4z4*

Description

Macrosatellite repeats D4Z4 in the subtelomere of chromosome 4q.
Locus (hg18): 4q35.2
Unit (kb): 3.3
Restriction enzyme: EcoRI + HindIII/EcoRI + BlnI/XapI
Encoded product : DUX4

Usage

```
d4z4
```

Format

A vector of counts with 410 elements.

References

Schaap M, Lemmers RJ, Maassen R, van der Vliet PJ, Hoogerheide LF, van Dijk HK, Basturk N, de Knijff P, van der Maarel SM (2013). "Genome-wide analysis of macrosatellite repeat copy number variation in worldwide populations: evidence for differences and commonalities in size distributions and size restrictions." *BMC Genomics*, **14**(1), 143. ISSN 1471-2164, doi:10.1186/1471216414143.

`discrete_MF` *Mode-finding algorithm for mixture of discrete distributions*

Description

Function to estimate modes in mixtures of discrete distributions following; see Cross et al. (2023).

Usage

```
discrete_MF(
  mcmc,
  data,
  pars_names,
  dist = "NA",
  pmf_func = NULL,
  type = "all",
  show_plot = FALSE
)
```

Arguments

`mcmc`	Vector of estimated mixture parameters.
`data`	Vector of observations used for estimating the mixture.
`pars_names`	Names of the mixture parameters; first element should correspond to the mixture proportions.
`dist`	String indicating the distribution of the mixture components. Currently supports "poisson" and "shifted_poisson"; default is "NA"; only use this argument if you have used Poisson and shifted Poisson distributions identical to the one used in the package.
`pmf_func`	Pmf of the mixture components associated with the mcmc draws. (if mcmc estimation has not been carried out with **BayesMultiMode**); default is null.
`type`	Type of modes, either unique or all (the latter includes flat modes); default is "all".
`show_plot`	If true show the data and estimated modes; default is false.

Details

This algorithm returns the local maxima of the mixture

$$p(x) = \sum_{k=1}^{K} \pi_k p_k(x).$$

By definition, modes must satisfy either:

$$p_k(y_m - 1) < p_k(y_m) > p_k(y_m + 1)$$

;

$$p_k(y_m - 1) < p_k(y_m) = p_k(y_m + 1) = \ldots = p_k(y_m + l - 1) > p_k(y_m + l).$$

The algorithm evaluate each location point with these two conditions.

Value

Vector of estimated modes.

References

Cross JL, Hoogerheide L, Labonne P, van Dijk HK (2023). "Credible Mode Determination in Multimodal Economic and Financial Data Distributions." *Tinbergen Institute Discussion Paper TI 2023-038/III*.

Schaap M, Lemmers RJ, Maassen R, van der Vliet PJ, Hoogerheide LF, van Dijk HK, Basturk N, de Knijff P, van der Maarel SM (2013). "Genome-wide analysis of macrosatellite repeat copy number variation in worldwide populations: evidence for differences and commonalities in size distributions and size restrictions." *BMC Genomics*, **14**(1), 143. ISSN 1471-2164, doi:10.1186/1471216414143.

Examples

```
# Example with the poisson distribution ===================================
lambda = c(0.1,10)
p = c(0.5,0.5)
params = c(eta = p, lambda = lambda)
pars_names = c("eta", "lambda")
dist = "poisson"

data = c(rpois(p[1]*1e3, lambda[1]),
         rpois(p[2]*1e3, lambda[2]))

modes = discrete_MF(params, data = data, pars_names = pars_names, dist = dist)

# Example with an arbitrary distribution ==================================
mu = c(20,5)
size = c(20,0.5)
p = c(0.5,0.5)
params = c(eta = p, mu = mu, size = size)
pars_names = c("eta", "mu", "size")

data = c(rnbinom(p[1]*1e3, mu = mu[1], size = size[1]),
         rnbinom(p[2]*1e3, mu = mu[2], size = size[2]))

pmf_func <- function(x, pars) {
  dnbinom(x, mu = pars["mu"], size = pars["size"])
}

modes = discrete_MF(params, data = data, pars_names = pars_names, pmf_func = pmf_func)
```

`fixed_point` *Modal fixed-point algorithm*

Description

Algorithm for estimating modes in mixture of Normal distributions from Carreira-Perpinan (2000).

Usage

```
fixed_point(
  mcmc,
  data,
  pars_names,
  tol_x = sd(data)/10,
  tol_conv = 1e-08,
  show_plot = F
)
```

Arguments

`mcmc`	Vector of estimated mixture parameters.
`data`	Vector of observations used for estimating the mixture.
`pars_names`	Names of the mixture parameters; first element should correspond to the mixture proportions; second to the mean; third to the standard deviation.
`tol_x`	Tolerance parameter for distance in-between modes; default is sd(data)/10; if two modes are closer than `tol_x`, only the first estimated mode is kept.
`tol_conv`	Tolerance parameter for convergence of the algorithm; default is 1e-8.
`show_plot`	If true show the data and estimated modes; default is false.

Details

This algorithm returns the local maxima of the mixture

$$p(x) = \sum_{k=1}^{K} \pi_k p_k(x),$$

where p_k comes from the Normal family. Following Carreira-perpinan (2000), a mode x is found by iterating the two steps:

$$(i) \quad p(k|x^{(n)}) = \frac{\pi_k p_k(x^{(n)})}{p(x^{(n)})},$$

$$(ii) \quad x^{(n+1)} = f(x^{(n)}),$$

with

$$f(x) = (\sum_k p(k|x)\sigma_k)^{-1} \sum_k p(k|x)\sigma_k \mu_k,$$

until convergence, that is, until $abs(x^{(n+1)} - x^{(n)}) < \text{tol}_{\text{conv}}$, where tol_{conv} is an argument with default value $1e{-}8$. Following Carreira-perpinan (2000), the algorithm is started at each component location. Separately, it is necessary to identify identical modes which diverge only up to a small value. By default modes which are closer than $sd(y)/10$ are merged; this tolerance value can be controlled with the argument `tol_x`.

Value

Vector of estimated modes.

References

Carreira-Perpinan MA (2000). "Mode-finding for mixtures of Gaussian distributions." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(11), 1318–1323. ISSN 1939-3539, doi:10.1109/34.888716, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Examples

```
mu = c(0,5)
sigma = c(1,2)
p = c(0.5,0.5)

data = c(rnorm(p[1]*100, mu[1], sigma[1]), rnorm(p[2]*100, mu[2], sigma[2]))
params = c(eta = p, mu = mu, sigma = sigma)
pars_names = c("eta", "mu", "sigma")
modes = fixed_point(params, data, pars_names)
```

`galaxy` *Galaxy series*

Description

Velocity at which 82 galaxies in the Corona Borealis region are moving away from our galaxy, scaled by 1000.

Usage

```
galaxy
```

Format

An object of class `numeric` of length 82.

Source

https://people.maths.bris.ac.uk/~mapjg/mixdata

References

Richardson S, Green PJ (1997). "On Bayesian Analysis of Mixtures with an Unknown Number of Components." *Journal of the Royal Statistical Society. Series B (Methodological)*, **59**(4), pp. 731–792. ISSN 00359246.

`MEM` *Modal EM algorithm (MEM)*

Description

Algorithm from Li and Lindsay (2007) to find modes in mixture of continuous distributions.

Usage

```
MEM(
  mcmc,
  data,
  pars_names,
  dist = "NA",
  pdf_func = NULL,
  tol_x = sd(data)/10,
  tol_conv = 1e-08,
  show_plot = FALSE
)
```

Arguments

`mcmc`	Vector of estimated mixture parameters.
`data`	Vector of observations used for estimating the mixture.
`pars_names`	Names of the mixture parameters; the first element of this vector should be the name of the mixture proportions. If you have used the skew normal of Azzalini, then the second element should correspond to the location, the third to the scale and the fourth to the shape.
`dist`	String indicating the distribution of the mixture components; default is "NA". Currently supports "normal" and "skew_normal"; not needed if pdf_func is provided.
`pdf_func`	Pdf of the mixture components associated with the mcmc draws (if mcmc estimation has not been carried out with BayesMultiMode); default is null.
`tol_x`	Tolerance parameter for distance in-between modes; default is sd(data)/10; if two modes are closer than `tol_x`, only the first estimated mode is kept.
`tol_conv`	Tolerance parameter for convergence of the algorithm; default is 1e-8.
`show_plot`	If true show the data and estimated modes; default is false

Details

This algorithm returns the local maxima of the mixture

$$p(x) = \sum_{k=1}^{K} \pi_k p_k(x),$$

where p_k is a density function. Following Li and Lindsay (2007), a mode x is found by iterating the two steps:

$$(i) \quad p(k|x^{(n)}) = \frac{\pi_k p_k(x^{(n)})}{p(x^{(n)})},$$

$$(ii) \quad x^{(n+1)} = \text{argmax}_x \sum_k p(k|x) \text{log} p_k(x^{(n)}),$$

until convergence, that is, until $abs(x^{(n+1)} - x^{(n)}) < \text{tol}_{\text{conv}}$, where tol_{conv} is an argument with default value $1e-8$. The algorithm is started at each component location. Separately, it is necessary to identify identical modes which diverge only up to a small value. By default modes which are closer than $sd(y)/10$ are merged; this tolerance value can be controlled with the argument `tol_x`.

While it is also possible to use the MEM algorithm for Normal mixtures, this is not recommended because the algorithm is less efficient than the fixed-point method in this particular case.

Value

Vector of estimated modes.

References

Li J, Ray S, Lindsay BG (2007). "A Nonparametric Statistical Approach to Clustering via Mode Identification." *Journal of Machine Learning Research*, **8**, 1687-1723.

Examples

```
# Example with the skew normal ================================================
xi = c(0,6)
omega = c(1,2)
alpha = c(0,0)
p = c(0.8,0.2)
params = c(eta = p, xi = xi, omega = omega, alpha = alpha)
pars_names = c("eta", "xi", "omega", "alpha")
dist = "skew_normal"

data = c(sn::rsn(p[1]*100, xi[1], omega[1], alpha[1]),
         sn::rsn(p[2]*100, xi[2], omega[2], alpha[2]))

modes = MEM(params, data = data, pars_names = pars_names, dist = dist)

# Example with an arbitrary distribution ======================================
```

```
xi = c(0,6)
omega = c(1,2)
alpha = c(0,0)
nu = c(3,100)
p = c(0.8,0.2)
params = c(eta = p, mu = xi, sigma = omega, xi = alpha, nu = nu)
pars_names = c("eta", "mu", "sigma", "xi", "nu")

pdf_func <- function(x, pars) {
  sn::dst(x, pars["mu"], pars["sigma"], pars["xi"], pars["nu"])
}

data = c(sn::rst(p[1]*100, xi[1], omega[1], alpha = alpha[1], nu = nu[1]),
         sn::rst(p[2]*100, xi[2], omega[2], alpha = alpha[2], nu = nu[2]))

modes = MEM(params, pars_names = pars_names, data = data, pdf_func = pdf_func)
```

`new_BayesMixture` *Creating a S3 object of class* `BayesMixture`

Description

Function for creating an object of class `BayesMixture` which can subsequently be used as argument in [bayes_mode()]. This function is useful for users who want to use the mode inference functions of the package with MCMC output generated using other software packages.

Usage

```
new_BayesMixture(
  mcmc,
  data,
  K,
  burnin,
  dist = "NA",
  pars_names,
  pdf_func = NULL,
  dist_type
)
```

Arguments

`mcmc`	A matrix of MCMC draws.
`data`	A vector containing the data used for estimating the model and generating the MCMC draws.
`K`	Number of mixture components.
`burnin`	Number of draws to discard as burnin.

dist	Distribution family of the mixture components supported by the package (e.g. "normal", "student", "skew_normal" or "shifted_poisson").
pars_names	Names of the mixture parameters; first element should correspond to the mixture proportions.
pdf_func	Pdf or pmf of the mixture components; this input is used only if dist_name is invalid or NULL.
dist_type	Either "continuous" or "discrete".

Value

A list of class `BayesMixture` containing:

- data - Same as argument.
- dist_type - Same as argument.
- pars_names - Same as argument.
- mcmc - Matrix of MCMC draws where the rows corresponding to burnin have been discarded.
- mcmc_all - Original matrix of MCMC draws.

Examples

```
# Example with a Student t ================================================
mu = c(0.5,6)
sigma = c(1,2)
nu = c(5,5)
p = c(0.8,0.2)
params = c(eta = p, mu = mu, sigma = sigma, nu = nu)
pars_names = c("eta", "mu", "sigma", "nu")
dist_type = "continuous"

data = c(sn::rst(p[1]*1000, mu[1], sigma[1], nu = nu[1]),
         sn::rst(p[2]*1000, mu[2], sigma[2], nu = nu[2]))

fit = c(eta = p, mu = mu, sigma = sigma, nu = nu)
fit = rbind(fit, fit)

pdf_func = function(x, pars) {
  sn::dst(x, pars["mu"], pars["sigma"], pars["xi"], pars["nu"])
}

BM = new_BayesMixture(fit, data, K = 2, burnin = 1,
pars_names = pars_names, pdf_func = pdf_func, dist_type = dist_type)
```

plot.BayesMixture *Plot method for* BayesMixture *objects*

Description

Plot an estimated mixture for a given number of draws with a frequency distribution of the data.

Usage

```
## S3 method for class 'BayesMixture'
plot(x, max_size = 250, transparency = 0.1, ...)
```

Arguments

x	An object of class BayesMixture.
max_size	The number of MCMC draws to plot.
transparency	transparency of the density lines. Default is 0.1. Should be greater than 0 and below or equal to 1.
...	Not used.

plot.BayesMode *Plot method for* BayesMode *objects*

Description

Plot method for BayesMode objects

Usage

```
## S3 method for class 'BayesMode'
plot(x, graphs = c("p1", "number", "loc"), ...)
```

Arguments

x	An object of class BayesMode.
graphs	which plot to show ? Default is all three c("p1", "number", "loc").
...	Not used.

summary.BayesMode *Summary method for for* BayesMode *objects*

Description

Summary method for for BayesMode objects

Usage

```
## S3 method for class 'BayesMode'
summary(object, ...)
```

Arguments

object	An object of class BayesMode.
...	Not used.

Index