

Package ‘BayesMultiMode’

February 27, 2026

Type Package

Title Bayesian Mode Inference

Version 0.7.5

Description A two-step Bayesian approach for mode inference following Cross, Hoogerheide, Labonne and van Dijk (2024) <[doi:10.1016/j.econlet.2024.111579](https://doi.org/10.1016/j.econlet.2024.111579)>. First, a mixture distribution is fitted on the data using a sparse finite mixture (SFM) Markov chain Monte Carlo (MCMC) algorithm. 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 (>= 3.4.0), ggpubr, gtools, magrittr, MCMCglmm, mvtnorm, posterior, sn, stringr, tidy, Rdpack

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 7.3.3

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 <labonnepaul@gmail.com>

Repository CRAN

Date/Publication 2026-02-27 15:32:05 UTC

Contents

bayes_fit	2
bayes_mixture	7
bayes_mode	10
bayes_trace	13
ct47	14
cyclone	15
d4z4	16
galaxy	16
mixture	17
mix_mode	19
plot.bayes_mixture	22
plot.bayes_mode	23
plot.mixture	24
plot.mix_mode	24
print.bayes_mixture	25
print.bayes_mode	25
print.mixture	26
print.mix_mode	26
summary.bayes_mixture	27
summary.bayes_mode	27
summary.mixture	28
summary.mix_mode	28
Index	29

bayes_fit

Bayesian estimation of mixture distributions

Description

Estimation of a univariate mixture with unknown number of components using a sparse finite mixture Markov chain Monte Carlo (SFM MCMC) algorithm.

Usage

```

bayes_fit(
  data,
  K,
  dist,
  priors = list(),
  nb_iter = 2000,
  burnin = nb_iter/2,
  print = TRUE
)

```

Arguments

<code>data</code>	Vector of observations.
<code>K</code>	Maximum number of mixture components.
<code>dist</code>	String indicating the distribution of the mixture components; currently supports "normal", "skew_normal", "poisson" and "shifted_poisson".
<code>priors</code>	List of priors; default is an empty list which implies the following priors: $a_0 = 1,$ $A_0 = 200,$ $b_0 = \text{median}(y),$ $B_0 = (\max(y) - \min(y))^2$ (normal), $D_{xi} = 1,$ $D_{psi} = 1,$ (skew normal: $B_0 = \text{diag}(D_{xi}, D_{psi})$), $c_0 = 2.5,$ $l_0 = 1.1$ (poisson), $l_0 = 5$ (shifted poisson), $L_0 = 1.1/\text{median}(y),$ $L_0 = l_0 - 1$ (shifted poisson), $g_0 = 0.5,$ $G_0 = 100 * g_0 / c_0 / B_0$ (normal), $G_0 = g_0 / (0.5 * \text{var}(y))$ (skew normal).
<code>nb_iter</code>	Number of MCMC iterations; default is 2000.
<code>burnin</code>	Number of MCMC iterations used as burnin; default is <code>nb_iter/2</code> .
<code>print</code>	Showing MCMC progression ? Default is TRUE.

Details

Let $y_i, i = 1, \dots, n$ 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 $\pi_k \geq 0, k = 1, \dots, K$.

The exact number of components does not have to be known *a priori* when using an 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 estimation. Following Malsiner-Walli et al. (2016) a symmetric Dirichlet prior is used for the mixture weights:

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

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):

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

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

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

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 Frühwirth-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

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

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:

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

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

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

We set

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

and

$$\mathbf{B}_0 = \text{diag}(\mathbf{D}_{\text{xi}}, \mathbf{D}_{\text{psi}})$$

with $\mathbf{D}_{\text{xi}} = \mathbf{D}_{\text{psi}} = \mathbf{1}$.

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):

$$\lambda_k \sim \text{Gamma}(l_0, L_0).$$

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, see Cross et al. (2024).

Value

A list of class bayes_mixture containing:

data	Same as argument.
mcmc	Matrix of MCMC draws where the rows corresponding to burnin have been discarded;
mcmc_all	Matrix of MCMC draws.
loglik	Log likelihood at each MCMC draw.
K	Number of components.
dist	Same as argument.
pdf_func	The pdf/pmf of the mixture components.
dist_type	Type of the distribution, i.e. continuous or discrete.
pars_names	Names of the mixture components' parameters.
loc	Name of the location parameter of the mixture components.
nb_var	Number of variables/parameters in the mixture distribution.

References

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].

Cross JL, Hoogerheide L, Labonne P, van Dijk HK (2024). “Bayesian mode inference for discrete distributions in economics and finance.” *Economics Letters*, **235**, 111579. ISSN 0165-1765, doi:10.1016/j.econlet.2024.111579.

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.

Malsiner-Walli G, Frühwirth-Schnatter S, Grün 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.

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_fit(
  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_fit(
  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_fit(
  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_mixture

Creating a S3 object of class bayes_mixture

Description

Creates an object of class `bayes_mixture` which can subsequently be used as argument in `bayes_mode()`. This function is useful for users who want to use the mode inference capabilities of `BayesMultiMode` with mixture estimated using external software.

Usage

```

bayes_mixture(
  mcmc,
  data,
  burnin = 0,
  dist = NA_character_,

```

```

pdf_func = NULL,
dist_type = NA_character_,
loglik = NULL,
vars_to_keep = NA_character_,
vars_to_rename = NA_character_,
loc = NA_character_
)

```

Arguments

mcmc	A matrix of MCMC draws with one column per variable, e.g. eta1, eta2, ..., mu1, mu2, etc...
data	Vector of observation used for estimating the model.
burnin	Number of draws to discard as burnin; default is 0.
dist	Distribution family of the mixture components supported by the package (i.e. "normal", "student", "skew_normal" or "shifted_poisson"). If left unspecified, pdf_func is required.
pdf_func	(function) Pdf or pmf of the mixture components; this input is used only if dist is left unspecified. pdf_func should have two arguments : (i) the observation where the pdf is evaluated; (ii) a named vector representing the function parameters. For instance a normal pdf would take the form: pdf_func <- function(x, pars) dnorm(x, pars['mu'], pars['sigma']). The names of pars should correspond to variables in mcmc, e.g. "mu1", "mu2" etc...
dist_type	Either "continuous" or "discrete".
loglik	Vector showing the log likelihood at each MCMC draw.
vars_to_keep	(optional) Character vector containing the names of the variables to keep in mcmc, e.g. c("eta", "mu", "sigma").
vars_to_rename	(optional) Use for renaming variables/parameters in mcmc. A named character vector where the names are the new variable names and the elements the variables in mcmc, e.g. c("new_name" = "old_name").
loc	(for continuous mixtures other than Normal mixtures) String indicating the location parameter of the distribution; the latter is used to initialise the MEM algorithm.

Value

A list of class bayes_mixture containing:

data	Same as argument.
mcmc	Matrix of MCMC draws where the rows corresponding to burnin have been discarded;
mcmc_all	Matrix of MCMC draws.
loglik	Log likelihood at each MCMC draw.
K	Number of components.
dist	Same as argument.

pdf_func	The pdf/pmf of the mixture components.
dist_type	Type of the distribution, i.e. continuous or discrete.
pars_names	Names of the mixture components' parameters.
loc	Name of the location parameter of the mixture components.
nb_var	Number of parameters in the mixture distribution.

Examples

```
# Example with a Student t =====

# Constructing synthetic mcmc output
mu <- c(0.5, 6)
mu_mat <- matrix(rep(mu, 100) + rnorm(200, 0, 0.1),
  ncol = 2, byrow = TRUE
)

omega <- c(1, 2)
sigma_mat <- matrix(rep(omega, 100) + rnorm(200, 0, 0.1),
  ncol = 2, byrow = TRUE
)

nu <- c(5, 5)
nu_mat <- matrix(rep(nu, 100) + rnorm(200, 0, 0.1),
  ncol = 2, byrow = TRUE
)

eta <- c(0.8, 0.2)
eta_mat <- matrix(rep(eta[1], 100) + rnorm(100, 0, 0.05),
  ncol = 1
)
eta_mat <- cbind(eta_mat, 1 - eta_mat)

xi_mat <- matrix(0, 100, 2)

fit <- cbind(eta_mat, mu_mat, sigma_mat, nu_mat, xi_mat)
colnames(fit) <- c(
  "eta1", "eta2", "mu1", "mu2",
  "omega1", "omega2", "nu1", "nu2", "xi1", "xi2"
)

# sampling observations
data <- c(
  sn::rst(eta[1] * 1000, mu[1], omega[1], nu = nu[1]),
  sn::rst(eta[2] * 1000, mu[2], omega[2], nu = nu[2])
)

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

dist_type <- "continuous"
```

```

BM <- bayes_mixture(fit, data,
  burnin = 50,
  pdf_func = pdf_func, dist_type = dist_type,
  vars_to_rename = c("sigma" = "omega"), loc = "xi"
)
# plot(BM)

```

 bayes_mode

Bayesian mode inference

Description

Bayesian inference on the modes in a univariate mixture estimated with MCMC methods, see Cross et al. (2024). Provides posterior probabilities of the number of modes and their locations. Under the hood it calls the function `mix_mode()` to find the modes in each MCMC draw.

Usage

```

bayes_mode(
  BayesMix,
  rd = 1,
  tol_mixp = 0,
  tol_x = sd(BayesMix$data)/10,
  tol_conv = 1e-08,
  inside_range = TRUE,
  range = c(min(BayesMix$data), max(BayesMix$data)),
  conditional_nb_modes = NULL
)

```

Arguments

BayesMix	An object of class <code>bayes_mixture</code> generated with either <code>bayes_fit()</code> or <code>bayes_mixture()</code> .
rd	(for continuous mixtures) Integer indicating the number of decimal places when rounding the distribution's support. It is necessary to compute posterior probabilities of mode locations.
tol_mixp	Components with a mixture proportion below <code>tol_mixp</code> are discarded when estimating modes; note that this does not apply to the biggest component so that it is not possible to discard all components; should be between 0 and 1; default is 0.
tol_x	(for continuous mixtures) Tolerance parameter for distance in-between modes; default is <code>sd(data)/10</code> where <code>data</code> is the vector of observations from <code>BayesMix</code> . If two modes are closer than <code>tol_x</code> , only the first estimated mode is kept.
tol_conv	(for continuous mixtures) Tolerance parameter for convergence of the algorithm; default is <code>1e-8</code> .
inside_range	Should modes outside of range be discarded? Default is <code>TRUE</code> .

range limits of the support where modes are saved (if inside_range is TRUE);
 conditional_nb_modes Mcmc draws are filtered to include those with only conditional_nb_modes number of modes; default is $c(\min(\text{BayesMix}\$data), \max(\text{BayesMix}\$data))$. This sometimes occurs with very small components when K is large.

Details

Each draw from the MCMC output after burnin, $\theta^{(d)}$, $d = 1, \dots, D$, 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 $y_m^{(d)}$, $m = 1, \dots, M^{(d)}$, 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:

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

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

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

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 bayes_mode 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.
p_nb_modes	Matrix showing posterior probabilities for the number of modes.
p_mode_loc	Matrix showing posterior probabilities for mode locations.
mix_density	Mixture density at all mode locations in each draw.
algo	Algorithm used for mode estimation.
range	Range outside which modes are discarded if inside_range is TRUE.
conditional_nb_modes	From BayesMix.
BayesMix	BayesMix.

References

Cross JL, Hoogerheide L, Labonne P, van Dijk HK (2024). “Bayesian mode inference for discrete distributions in economics and finance.” *Economics Letters*, **235**, 111579. ISSN 0165-1765, [doi:10.1016/j.econlet.2024.111579](https://doi.org/10.1016/j.econlet.2024.111579).

Examples

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

# retrieve galaxy data
y <- galaxy

# estimation
bayesmix <- bayes_fit(
  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 DNA data
y <- d4z4

# estimation
bayesmix <- bayes_fit(
  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) #'
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, xi = c(0, 0))
fit <- rbind(fit, fit)

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

dist_type <- "continuous"

bayesmix <- bayes_mixture(fit, data,
  burnin = 1,
  pdf_func = pdf_func, dist_type = dist_type, loc = "mu"
)

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 bayes_mixture.
 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_fit(
  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](https://doi.org/10.1186/1471216414143).

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.

 mixture

 Creating a S3 object of class mixture

Description

Creates an object of class `mixture` which can subsequently be used as argument in `mix_mode()` for mode estimation.

Usage

```
mixture(
  pars,
  dist = NA_character_,
  pdf_func = NULL,
  dist_type = NA_character_,
  range,
  loc = NA_character_
)
```

Arguments

<code>pars</code>	Named vector of mixture parameters.
<code>dist</code>	Distribution family of the mixture components supported by the package (i.e. "normal", "student", "skew_normal" or "shifted_poisson"). If left unspecified, <code>pdf_func</code> is required.
<code>pdf_func</code>	(function) Pdf or pmf of the mixture components; this input is used only if <code>dist</code> is left unspecified. <code>pdf_func</code> should have two arguments : (i) the observation where the pdf is evaluated; (ii) a named vector representing the function parameters. For instance a normal pdf would take the form: <code>pdf_func <- function(x, par) dnorm(x, par['mu'], par['sigma'])</code> . The names of <code>par</code> should correspond to variables in <code>pars</code> , e.g. "mu1", "mu2" etc...
<code>dist_type</code>	Type of the distribution, either "continuous" or "discrete".
<code>range</code>	upper and lower limit of the range where the mixture should be evaluated.
<code>loc</code>	(for continuous mixtures other than Normal mixtures) String indicating the location parameter of the distribution; the latter is used to initialise the MEM algorithm.

Value

A list of class mixture containing:

<code>pars</code>	Same as argument.
<code>pars_names</code>	Names of the parameters of the components' distribution.
<code>dist</code>	Same as argument.
<code>pdf_func</code>	Pdf (or pmf) of the mixture components.
<code>dist_type</code>	Same as argument.
<code>loc</code>	Type of the distribution, either "continuous" or "discrete".
<code>nb_var</code>	Number of parameters in the mixture distribution.
<code>K</code>	Number of mixture components.
<code>range</code>	Same as argument.

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)
dist <- "skew_normal"

mix <- mixture(params, dist = dist, range = c(-2, 10))

# summary(mix)
# plot(mix)

# Example with an arbitrary distribution =====
mu <- c(0, 6)
omega <- c(1, 2)
xi <- c(0, 0)
nu <- c(3, 100)
p <- c(0.8, 0.2)
params <- c(eta = p, mu = mu, sigma = omega, xi = xi, nu = nu)

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

mix <- mixture(params,
  pdf_func = pdf_func,
  dist_type = "continuous", loc = "mu", range = c(-2, 10)
)

# summary(mix)
# plot(mix, from = -4, to = 4)
```

mix_mode	<i>Mode estimation</i>
----------	------------------------

Description

Mode estimation in univariate mixture distributions. 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. (2024).

Usage

```

mix_mode(
  mixture,
  tol_mixp = 0,
  tol_x = 1e-06,
  tol_conv = 1e-08,
  type = "all",
  inside_range = TRUE
)

```

Arguments

mixture	An object of class mixture generated with <code>mixture()</code> .
tol_mixp	Components with a mixture proportion below <code>tol_mixp</code> are discarded when estimating modes; note that this does not apply to the biggest component so that it is not possible to discard all components; should be between 0 and 1; default is 0.
tol_x	(for continuous mixtures) Tolerance parameter for distance in-between modes; default is 1e-6; if two modes are closer than <code>tol_x</code> the first estimated mode is kept.
tol_conv	(for continuous mixtures) Tolerance parameter for convergence of the algorithm; default is 1e-8.
type	(for discrete mixtures) Type of modes, either "unique" or "all" (the latter includes flat modes); default is "all".
inside_range	Should modes outside of <code>mixture\$range</code> be discarded? Default is TRUE. This sometimes occurs with very small components when K is large.

Details

This function finds modes in a univariate mixture defined as:

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

where p_k is a density or probability mass/density function.

Fixed-point algorithm 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

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

until convergence, that is, until $abs(x^{(n+1)} - x^{(n)}) < tol_{conv}$, 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; this tolerance value can be controlled with the argument `tol_x`.

MEM algorithm Following Li et al. (2007), a mode x is found by iterating the two steps:

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

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

until convergence, that is, until $abs(x^{(n+1)} - x^{(n)}) < tol_{conv}$, 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. Modes which are closer then `tol_x` are merged.

Discrete method By definition, modes must satisfy either:

$$p(y_m - 1) < p(y_m) > p(y_m + 1);$$

$$p(y_m - 1) < p(y_m) = p(y_m + 1) = \dots = p(y_m + l - 1) > p(y_m + l).$$

The algorithm evaluate each location point with these two conditions.

Value

A list of class `mix_mode` containing:

<code>mode_estimates</code>	estimates of the mixture modes.
<code>algo</code>	algorithm used for mode estimation.
<code>dist</code>	from mixture.
<code>dist_type</code>	type of mixture distribution, i.e. continuous or discrete.
<code>pars</code>	from mixture.
<code>pdf_func</code>	from mixture.
<code>K</code>	from mixture.
<code>nb_var</code>	from mixture.

References

Cross JL, Hoogerheide L, Labonne P, van Dijk HK (2024). “Bayesian mode inference for discrete distributions in economics and finance.” *Economics Letters*, **235**, 111579. ISSN 0165-1765, doi:10.1016/j.econlet.2024.111579.

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.

Cross JL, Hoogerheide L, Labonne P, van Dijk HK (2024). “Bayesian mode inference for discrete distributions in economics and finance.” *Economics Letters*, **235**, 111579. ISSN 0165-1765, doi:10.1016/j.econlet.2024.111579.

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 a normal distribution =====
mu <- c(0, 5)
sigma <- c(1, 2)
p <- c(0.5, 0.5)

params <- c(eta = p, mu = mu, sigma = sigma)
mix <- mixture(params, dist = "normal", range = c(-5, 15))
modes <- mix_mode(mix)

# summary(modes)
# plot(modes)

# Example with a 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)
dist <- "skew_normal"

mix <- mixture(params, dist = dist, range = c(-5, 15))
modes <- mix_mode(mix)
# summary(modes)
# plot(modes)

# Example with an arbitrary continuous 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)

pdf_func <- function(x, pars) {
```

```

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

mix <- mixture(params,
  pdf_func = pdf_func,
  dist_type = "continuous", loc = "mu", range = c(-5, 15)
)
modes <- mix_mode(mix)

# summary(modes)
# plot(modes, from = -4, to = 4)

# Example with a poisson distribution =====
lambda <- c(0.1, 10)
p <- c(0.5, 0.5)
params <- c(eta = p, lambda = lambda)
dist <- "poisson"

mix <- mixture(params, range = c(0, 50), dist = dist)

modes <- mix_mode(mix)

# summary(modes)
# plot(modes)

# Example with an arbitrary discrete distribution =====
mu <- c(20, 5)
size <- c(20, 0.5)
p <- c(0.5, 0.5)
params <- c(eta = p, mu = mu, size = size)

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

mix <- mixture(params,
  range = c(0, 50),
  pdf_func = pmf_func, dist_type = "discrete"
)
modes <- mix_mode(mix)

# summary(modes)
# plot(modes)

```

Description

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

Usage

```
## S3 method for class 'bayes_mixture'
plot(x, draws = 250, draw = NULL, bins = 30, alpha = 0.1, ...)
```

Arguments

x	An object of class bayes_mixture.
draws	The number of MCMC draws to plot.
draw	Plot estimated mixture in draw draw; note that draws is discarded. Default is NULL.
bins	(for continuous mixtures) Number of bins for the histogram of the data. Passed to geom_histogram().
alpha	transparency of the density lines. Default is 0.1. Should be greater than 0 and below or equal to 1.
...	Not used.

plot.bayes_mode	<i>Plot method for bayes_mode objects</i>
-----------------	---

Description

Plot method for bayes_mode objects

Usage

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

Arguments

x	An object of class bayes_mode.
graphs	which plot to show ? Default is all three c("p1", "number", "loc").
draw	Plot modes in a given mcmc draw; note that graphs is discarded. Default is NULL.
...	Not used.

plot.mixture	<i>Plot method for mixture objects</i>
--------------	--

Description

Plot method for mixture objects

Usage

```
## S3 method for class 'mixture'
plot(x, from = x$range[1], to = x$range[2], ...)
```

Arguments

x	An object of class mixture.
from	the lower limit of the range over which the function will be plotted. Default is x\$range[1].
to	the upper limit of the range over which the function will be plotted. Default is x\$range[2].
...	Not used.

plot.mix_mode	<i>Plot method for mix_mode objects</i>
---------------	---

Description

Plot method for mix_mode objects

Usage

```
## S3 method for class 'mix_mode'
plot(x, from = x$range[1], to = x$range[2], ...)
```

Arguments

x	An object of class mix_mode.
from	the lower limit of the range over which the function will be plotted. Default is x\$range[1].
to	the upper limit of the range over which the function will be plotted. Default is x\$range[2].
...	Not used.

print.bayes_mixture *Print method for bayes_mixture objects*

Description

Print method for bayes_mixture objects

Usage

```
## S3 method for class 'bayes_mixture'  
print(x, max_length = 6L, max_width = 6L, print_all = F, ...)
```

Arguments

x	An object of class bayes_mixture.
max_length	maximum number of elements (for vector) or rows (for matrices) to show. Default is 6L.
max_width	maximum number of columns to show (for matrices). Default is 6L.
print_all	override max_length and max_width to print everything? Default is FALSE.
...	Not used.

print.bayes_mode *Print method for bayes_mode objects*

Description

Print method for bayes_mode objects

Usage

```
## S3 method for class 'bayes_mode'  
print(x, max_length = 6L, max_width = 6L, print_all = F, ...)
```

Arguments

x	An object of class bayes_mode.
max_length	maximum number of elements (for vector) or rows (for matrices) to show. Default is 6L.
max_width	maximum number of columns to show (for matrices). Default is 6L.
print_all	override max_length and max_width to print everything? Default is FALSE.
...	Not used.

print.mixture	<i>Print method for mixture objects</i>
---------------	---

Description

Print method for mixture objects

Usage

```
## S3 method for class 'mixture'
print(x, max_length = 6L, max_width = 6L, print_all = F, ...)
```

Arguments

x	An object of class mixture.
max_length	maximum number of elements (for vector) or rows (for matrices) to show. Default is 6L.
max_width	maximum number of columns to show (for matrices). Default is 6L.
print_all	override max_length and max_width to print everything? Default is FALSE.
...	Not used.

print.mix_mode	<i>Print method for mix_mode objects</i>
----------------	--

Description

Print method for mix_mode objects

Usage

```
## S3 method for class 'mix_mode'
print(x, max_length = 6L, max_width = 6L, print_all = F, ...)
```

Arguments

x	An object of class mix_mode.
max_length	maximum number of elements (for vector) or rows (for matrices) to show. Default is 6L.
max_width	maximum number of columns to show (for matrices). Default is 6L.
print_all	override max_length and max_width to print everything? Default is FALSE.
...	Not used.

summary.bayes_mixture *Summary method for bayes_mixture objects The summary of MCMC draws is given by the function summarise_draws from package **posterior**.*

Description

Summary method for bayes_mixture objects The summary of MCMC draws is given by the function summarise_draws from package **posterior**.

Usage

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

Arguments

object	An object of class bayes_mixture.
...	Not used.

summary.bayes_mode *Summary method for bayes_mode objects*

Description

Summary method for bayes_mode objects

Usage

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

Arguments

object	An object of class bayes_mode.
...	Not used.

summary.mixture	<i>Summary method for mixture objects</i>
-----------------	---

Description

Summary method for mixture objects

Usage

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

Arguments

object	An object of class mixture.
...	Not used.

summary.mix_mode	<i>Summary method for mix_mode objects</i>
------------------	--

Description

Summary method for mix_mode objects

Usage

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

Arguments

object	An object of class mix_mode.
...	Not used.

Index

* datasets

- ct47, [14](#)
- cyclone, [15](#)
- d4z4, [16](#)
- galaxy, [16](#)

- bayes_fit, [2](#)
- bayes_fit(), [10](#)
- bayes_mixture, [7](#)
- bayes_mixture(), [10](#)
- bayes_mode, [10](#)
- bayes_mode(), [7](#)
- bayes_trace, [13](#)
- bayesplot::mcmc_trace(), [13](#), [14](#)

- ct47, [14](#)
- cyclone, [15](#)

- d4z4, [16](#)

- galaxy, [16](#)

- mix_mode, [19](#)
- mix_mode(), [10](#), [17](#)
- mixture, [17](#)
- mixture(), [19](#)

- plot.bayes_mixture, [22](#)
- plot.bayes_mode, [23](#)
- plot.mix_mode, [24](#)
- plot.mixture, [24](#)
- print.bayes_mixture, [25](#)
- print.bayes_mode, [25](#)
- print.mix_mode, [26](#)
- print.mixture, [26](#)

- summary.bayes_mixture, [27](#)
- summary.bayes_mode, [27](#)
- summary.mix_mode, [28](#)
- summary.mixture, [28](#)