

# Package: **bmixture** (via r-universe)

October 9, 2024

**Title** Bayesian Estimation for Finite Mixture of Distributions

**Version** 1.7

**Description** Provides statistical tools for Bayesian estimation of mixture distributions, mainly a mixture of Gamma, Normal, and t-distributions. The package is implemented based on the Bayesian literature for the finite mixture of distributions, including Mohammadi and et al. (2013) <[doi:10.1007/s00180-012-0323-3](https://doi.org/10.1007/s00180-012-0323-3)> and Mohammadi and Salehi-Rad (2012) <[doi:10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)>.

**URL** <https://www.uva.nl/profile/a.mohammadi>

**Depends** R (>= 3.0.0)

**Imports** BDgraph

**License** GPL (>= 2)

**NeedsCompilation** yes

**Author** Reza Mohammadi [aut, cre]  
(<<https://orcid.org/0000-0001-9538-0648>>)

**Maintainer** Reza Mohammadi <a.mohammadi@uva.nl>

**Date/Publication** 2021-05-11 11:42:19 UTC

**Repository** <https://rezamoammadi.r-universe.dev>

**RemoteUrl** <https://github.com/cran/bmixture>

**RemoteRef** HEAD

**RemoteSha** 2632fd6b8c9c7f2471d815e4f13f50027d5b4f73

## Contents

bmixture-package . . . . .	2
bmixgamma . . . . .	3
bmixnorm . . . . .	6
bmixt . . . . .	9
galaxy . . . . .	12
mixgamma . . . . .	13

mixnorm . . . . .	14
mixt . . . . .	16
plot.bmixgamma . . . . .	17
plot.bmixnorm . . . . .	18
plot.bmixt . . . . .	19
print.bmixgamma . . . . .	20
print.bmixnorm . . . . .	21
print.bmixt . . . . .	22
rdirichlet . . . . .	23
summary.bmixgamma . . . . .	24
summary.bmixnorm . . . . .	25
summary.bmixt . . . . .	26

<b>Index</b>	<b>28</b>
--------------	-----------

---

bmixture-package	<i>Bayesian Estimation for Finite Mixture of Distributions</i>
------------------	--

---

## Description

The R package **bmixture** provides statistical tools for Bayesian estimation in finite mixture of distributions. The package implemented the improvements in the Bayesian literature, including Mohammadi and Salehi-Rad (2012) and Mohammadi et al. (2013). Besides, the package contains several functions for simulation and visualization, as well as a real dataset taken from the literature.

## How to cite this package

Whenever using this package, please cite as Mohammadi R. (2019). **bmixture**: Bayesian Estimation for Finite Mixture of Distributions, R package version 1.5, <https://CRAN.R-project.org/package=bmixture>

## Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

## References

- Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)
- Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)
- Stephens, M. (2000) Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. *Annals of statistics*, 28(1):40-74, doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364)
- Richardson, S. and Green, P. J. (1997) On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: series B*, 59(4):731-792, doi: [10.1111/14679868.00095](https://doi.org/10.1111/14679868.00095)

Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711-732, doi: [10.1093/biomet/82.4.711](https://doi.org/10.1093/biomet/82.4.711)

Cappe, O., Christian P. R., and Tobias, R. (2003) Reversible jump, birth and death and more general continuous time Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 65(3):679-700

Wade, S. and Ghahramani, Z. (2018) Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). *Bayesian Analysis*, 13(2):559-626, doi: [10.1214/17BA1073](https://doi.org/10.1214/17BA1073)

## Examples

```
## Not run:

require( bmixture )

data( galaxy )

# Running bdmcmc algorithm for the galaxy dataset
mcmc_sample = bmixnorm( data = galaxy )

summary( mcmc_sample )
plot( mcmc_sample )
print( mcmc_sample)

# simulating data from mixture of Normal with 3 components
n      = 500
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )
weight = c( 0.3, 0.5, 0.2 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Running bdmcmc algorithm for the above simulation data set
bmixnorm.obj = bmixnorm( data, k = 3, iter = 1000 )

summary( bmixnorm.obj )

## End(Not run)
```

**Description**

This function consists of several sampling algorithms for Bayesian estimation for a mixture of Gamma distributions.

**Usage**

```
bmixgamma( data, k = "unknown", iter = 1000, burnin = iter / 2, lambda = 1,
           mu = NULL, nu = NULL, kesi = NULL, tau = NULL, k.start = NULL,
           alpha.start = NULL, beta.start = NULL, pi.start = NULL,
           k.max = 30, trace = TRUE )
```

**Arguments**

data	vector of data with size n.
k	number of components of mixture distribution. It can take an integer values.
iter	number of iteration for the sampling algorithm.
burnin	number of burn-in iteration for the sampling algorithm.
lambda	For the case k = "unknown", it is the parameter of the prior distribution of number of components k.
mu	parameter of alpha in mixture distribution.
nu	parameter of alpha in mixture distribution.
kesi	parameter of beta in mixture distribution.
tau	parameter of beta in mixture distribution.
k.start	For the case k = "unknown", initial value for number of components of mixture distribution.
alpha.start	Initial value for parameter of mixture distribution.
beta.start	Initial value for parameter of mixture distribution.
pi.start	Initial value for parameter of mixture distribution.
k.max	For the case k = "unknown", maximum value for the number of components of mixture distribution.
trace	Logical: if TRUE (default), tracing information is printed.

**Details**

Sampling from finite mixture of Gamma distribution, with density:

$$Pr(x|k, \underline{\pi}, \underline{\alpha}, \underline{\beta}) = \sum_{i=1}^k \pi_i \text{Gamma}(x|\alpha_i, \beta_i),$$

where k is the number of components of mixture distribution (as a default we assume is unknown) and

$$\text{Gamma}(x|\alpha_i, \beta_i) = \frac{(\beta_i)^{\alpha_i}}{\Gamma(\alpha_i)} x^{\alpha_i-1} e^{-\beta_i x}.$$

The prior distributions are defined as below

$$P(K = k) \propto \frac{\lambda^k}{k!}, \quad k = 1, \dots, k_{max},$$

$$\pi_i | k \sim \text{Dirichlet}(1, \dots, 1),$$

$$\alpha_i | k \sim \text{Gamma}(\nu, \nu),$$

$$\beta_i | k \sim \text{Gamma}(\eta, \tau),$$

for more details see Mohammadi et al. (2013), doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233).

### Value

An object with S3 class "bmixgamma" is returned:

all_k	a vector which includes the waiting times for all iterations. It is needed for monitoring the convergence of the BD-MCMC algorithm.
all_weights	a vector which includes the waiting times for all iterations. It is needed for monitoring the convergence of the BD-MCMC algorithm.
pi_sample	a vector which includes the MCMC samples after burn-in from parameter pi of mixture distribution.
alpha_sample	a vector which includes the MCMC samples after burn-in from parameter alpha of mixture distribution.
beta_sample	a vector which includes the MCMC samples after burn-in from parameter beta of mixture distribution.
data	original data.

### Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

### References

- Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)
- Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)
- Stephens, M. (2000) Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. *Annals of statistics*, 28(1):40-74, doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364)
- Richardson, S. and Green, P. J. (1997) On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: series B*, 59(4):731-792, doi: [10.1111/14679868.00095](https://doi.org/10.1111/14679868.00095)
- Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711-732, doi: [10.1093/biomet/82.4.711](https://doi.org/10.1093/biomet/82.4.711)

Cappe, O., Christian P. R., and Tobias, R. (2003) Reversible jump, birth and death and more general continuous time Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 65(3):679-700

Wade, S. and Ghahramani, Z. (2018) Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). *Bayesian Analysis*, 13(2):559-626, doi: [10.1214/17BA1073](https://doi.org/10.1214/17BA1073)

### See Also

[bmixnorm](#), [bmixt](#), [bmixgamma](#)

### Examples

```
## Not run:
set.seed( 70 )

# simulating data from mixture of gamma with two components
n      = 1000 # number of observations
weight = c( 0.6, 0.4 )
alpha  = c( 12 , 1 )
beta   = c( 3 , 2 )

data = rmixgamma( n = n, weight = weight, alpha = alpha, beta = beta )

# plot for simulation data
hist( data, prob = TRUE, nclass = 50, col = "gray" )

x      = seq( 0, 10, 0.05 )
truth  = dmixgamma( x, weight, alpha, beta )

lines( x, truth, lwd = 2 )

# Running bdmcmc algorithm for the above simulation data set
bmixgamma.obj = bmixgamma( data, iter = 1000 )

summary( bmixgamma.obj )

plot( bmixgamma.obj )

## End(Not run)
```

---

bmixnorm

*Sampling algorithm for mixture of Normal distributions*


---

### Description

This function consists of several sampling algorithms for Bayesian estimation for finite a mixture of Normal distributions.

**Usage**

```
bmixnorm( data, k = "unknown", iter = 1000, burnin = iter / 2, lambda = 1,
          k.start = NULL, mu.start = NULL, sig.start = NULL, pi.start = NULL,
          k.max = 30, trace = TRUE )
```

**Arguments**

data	vector of data with size n.
k	number of components of mixture distribution. It can take an integer values.
iter	number of iteration for the sampling algorithm.
burnin	number of burn-in iteration for the sampling algorithm.
lambda	For the case k = "unknown", it is the parameter of the prior distribution of number of components k.
k.start	For the case k = "unknown", initial value for number of components of mixture distribution.
mu.start	Initial value for parameter of mixture distribution.
sig.start	Initial value for parameter of mixture distribution.
pi.start	Initial value for parameter of mixture distribution.
k.max	For the case k = "unknown", maximum value for the number of components of mixture distribution.
trace	Logical: if TRUE (default), tracing information is printed.

**Details**

Sampling from finite mixture of Normal distribution, with density:

$$Pr(x|k, \underline{\pi}, \underline{\mu}, \underline{\sigma}) = \sum_{i=1}^k \pi_i N(x|\mu_i, \sigma_i^2),$$

where k is the number of components of mixture distribution (as a default we assume is unknown). The prior distributions are defined as below

$$P(K = k) \propto \frac{\lambda^k}{k!}, \quad k = 1, \dots, k_{max},$$

$$\pi_i|k \sim Dirichlet(1, \dots, 1),$$

$$\mu_i|k \sim N(\epsilon, \kappa),$$

$$\sigma_i|k \sim IG(g, h),$$

where IG denotes an inverted gamma distribution. For more details see for more details see Stephens, M. (2000), doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364).

**Value**

An object with S3 class "bmixnorm" is returned:

all_k	a vector which includes the waiting times for all iterations. It is needed for monitoring the convergence of the BD-MCMC algorithm.
all_weights	a vector which includes the waiting times for all iterations. It is needed for monitoring the convergence of the BD-MCMC algorithm.
pi_sample	a vector which includes the MCMC samples after burn-in from parameter pi of mixture distribution.
mu_sample	a vector which includes the MCMC samples after burn-in from parameter mu of mixture distribution.
sig_sample	a vector which includes the MCMC samples after burn-in from parameter sig of mixture distribution.
data	original data.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**References**

- Stephens, M. (2000) Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. *Annals of statistics*, 28(1):40-74, doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364)
- Richardson, S. and Green, P. J. (1997) On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: series B*, 59(4):731-792, doi: [10.1111/14679868.00095](https://doi.org/10.1111/14679868.00095)
- Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711-732, doi: [10.1093/biomet/82.4.711](https://doi.org/10.1093/biomet/82.4.711)
- Cappe, O., Christian P. R., and Tobias, R. (2003) Reversible jump, birth and death and more general continuous time Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 65(3):679-700
- Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)
- Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)
- Wade, S. and Ghahramani, Z. (2018) Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). *Bayesian Analysis*, 13(2):559-626, doi: [10.1214/17BA1073](https://doi.org/10.1214/17BA1073)

**See Also**

[bmixt](#), [bmixgamma](#), [rmixnorm](#)



**Examples**

```

## Not run:
data( galaxy )

set.seed( 70 )
# Runing bdmcmc algorithm for the galaxy dataset
mcmc_sample = bmixnorm( data = galaxy )

summary( mcmc_sample )
plot( mcmc_sample )
print( mcmc_sample)

# simulating data from mixture of Normal with 3 components
n      = 500
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixnorm.obj = bmixnorm( data, k = 3, iter = 1000 )

summary( bmixnorm.obj )

## End(Not run)

```

---

 bmixt

*Sampling algorithm for mixture of t-distributions*


---

**Description**

This function consists of several sampling algorithms for Bayesian estimation for finite mixture of t-distributions.

**Usage**

```

bmixt( data, k = "unknown", iter = 1000, burnin = iter / 2, lambda = 1, df = 1,
       k.start = NULL, mu.start = NULL, sig.start = NULL, pi.start = NULL,
       k.max = 30, trace = TRUE )

```

**Arguments**

data	vector of data with size n.
k	number of components of mixture distribution. Default is "unknown". It can take an integer values.
iter	number of iteration for the sampling algorithm.
burnin	number of burn-in iteration for the sampling algorithm.
lambda	For the case k = "unknown", it is the parameter of the prior distribution of number of components k.
df	Degrees of freedom (> 0, maybe non-integer). df = Inf is allowed.
k.start	For the case k = "unknown", initial value for number of components of mixture distribution.
mu.start	Initial value for parameter of mixture distribution.
sig.start	Initial value for parameter of mixture distribution.
pi.start	Initial value for parameter of mixture distribution.
k.max	For the case k = "unknown", maximum value for the number of components of mixture distribution.
trace	Logical: if TRUE (default), tracing information is printed.

**Details**

Sampling from finite mixture of t-distribution, with density:

$$Pr(x|k, \underline{\pi}, \underline{\mu}, \underline{\sigma}) = \sum_{i=1}^k \pi_i t_p(x|\mu_i, \sigma_i^2),$$

where k is the number of components of mixture distribution (as a default we assume is unknown). The prior distributions are defined as below

$$P(K = k) \propto \frac{\lambda^k}{k!}, \quad k = 1, \dots, k_{max},$$

$$\pi_i|k \sim Dirichlet(1, \dots, 1),$$

$$\mu_i|k \sim N(\epsilon, \kappa),$$

$$\sigma_i|k \sim IG(g, h),$$

where IG denotes an inverted gamma distribution. For more details see Stephens, M. (2000), doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364).

**Value**

An object with S3 class "bmixt" is returned:

all_k	a vector which includes the waiting times for all iterations. It is needed for monitoring the convergence of the BD-MCMC algorithm.
all_weights	a vector which includes the waiting times for all iterations. It is needed for monitoring the convergence of the BD-MCMC algorithm.
pi_sample	a vector which includes the MCMC samples after burn-in from parameter pi of mixture distribution.
mu_sample	a vector which includes the MCMC samples after burn-in from parameter mu of mixture distribution.
sig_sample	a vector which includes the MCMC samples after burn-in from parameter sig of mixture distribution.
data	original data.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**References**

- Stephens, M. (2000) Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. *Annals of statistics*, 28(1):40-74, doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364)
- Richardson, S. and Green, P. J. (1997) On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: series B*, 59(4):731-792, doi: [10.1111/14679868.00095](https://doi.org/10.1111/14679868.00095)
- Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711-732, doi: [10.1093/biomet/82.4.711](https://doi.org/10.1093/biomet/82.4.711)
- Cappe, O., Christian P. R., and Tobias, R. (2003) Reversible jump, birth and death and more general continuous time Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 65(3):679-700
- Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)
- Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)
- Wade, S. and Ghahramani, Z. (2018) Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). *Bayesian Analysis*, 13(2):559-626, doi: [10.1214/17BA1073](https://doi.org/10.1214/17BA1073)

**See Also**

[bmixnorm](#), [bmixgamma](#), [rmixt](#)

**Examples**

```
## Not run:
set.seed( 20 )

# simulating data from mixture of Normal with 3 components
n      = 2000
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixt.obj = bmixt( data, k = 3, iter = 5000 )

summary( bmixt.obj )

## End(Not run)
```

---

galaxy

*Galaxy data*


---

**Description**

This dataset considers of 82 observatons of the velocities (in 1000 km/second) of distant galaxies diverging from our own, from six well-separated conic sections of the Corona Borealis. The dataset has been analyzed under a variety of mixture models; See e.g. Stephens (2000).

**Usage**

```
data( galaxy )
```

**Format**

A data frame with 82 observations on the following variable.

speed a numeric vector giving the speed of galaxies (in 1000 km/second).

**References**

Stephens, M. (2000) Bayesian analysis of mixture models with an unknown number of components- an alternative to reversible jump methods. *Annals of statistics*, 28(1):40-74, doi: [10.1214/aos/1016120364](https://doi.org/10.1214/aos/1016120364)

**Examples**

```
data( galaxy )

hist( galaxy, prob = TRUE, xlim = c( 0, 40 ), ylim = c( 0, 0.3 ), nclass = 20,
      col = "gray", border = "white" )

lines( density( galaxy ), col = "black", lwd = 2 )
```

---

mixgamma	<i>Mixture of Gamma distribution</i>
----------	--------------------------------------

---

**Description**

Random generation and density function for a finite mixture of Gamma distribution.

**Usage**

```
rmixgamma( n = 10, weight = 1, alpha = 1, beta = 1 )

dmixgamma( x, weight = 1, alpha = 1, beta = 1 )
```

**Arguments**

n	number of observations.
x	vector of quantiles.
weight	vector of probability weights, with length equal to number of components ( $k$ ). This is assumed to sum to 1; if not, it is normalized.
alpha	vector of non-negative parameters of the Gamma distribution.
beta	vector of non-negative parameters of the Gamma distribution.

**Details**

Sampling from finite mixture of Gamma distribution, with density:

$$Pr(x|\underline{w}, \underline{\alpha}, \underline{\beta}) = \sum_{i=1}^k w_i \text{Gamma}(x|\alpha_i, \beta_i),$$

where

$$\text{Gamma}(x|\alpha_i, \beta_i) = \frac{(\beta_i)^{\alpha_i}}{\Gamma(\alpha_i)} x^{\alpha_i-1} e^{-\beta_i x}.$$

**Value**

Generated data as an vector with size  $n$ .

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

## References

Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)

Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)

## See Also

[rgamma](#), [rmixnorm](#), [rmixt](#)

## Examples

```
## Not run:
n      = 10000
weight = c( 0.6 , 0.3 , 0.1 )
alpha  = c( 100 , 200 , 300 )
beta   = c( 100/3, 200/4, 300/5 )

data = rmixgamma( n = n, weight = weight, alpha = alpha, beta = beta )

hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixgamma = dmixnorm( x, weight, alpha, beta )

lines( x, densmixgamma, lwd = 2 )

## End(Not run)
```

---

mixnorm

*Mixture of Normal distribution*

---

## Description

Random generation and density function for a finite mixture of univariate Normal distribution.

## Usage

```
rmixnorm( n = 10, weight = 1, mean = 0, sd = 1 )
```

```
dmixnorm( x, weight = 1, mean = 0, sd = 1 )
```

**Arguments**

n	number of observations.
x	vector of quantiles.
weight	vector of probability weights, with length equal to number of components ( $k$ ). This is assumed to sum to 1; if not, it is normalized.
mean	vector of means.
sd	vector of standard deviations.

**Details**

Sampling from finite mixture of Normal distribution, with density:

$$Pr(x|\underline{w}, \underline{\mu}, \underline{\sigma}) = \sum_{i=1}^k w_i N(x|\mu_i, \sigma_i).$$

**Value**

Generated data as an vector with size  $n$ .

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**References**

Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)

Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)

**See Also**

[rnorm](#), [rmixt](#), [rmixgamma](#)

**Examples**

```
## Not run:
n      = 10000
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

hist( data, prob = TRUE, nclass = 30, col = "gray" )
```

```
x          = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

## End(Not run)
```

---

mixt *Mixture of t-distribution*

---

### Description

Random generation and density function for a finite mixture of univariate t-distribution.

### Usage

```
rmixt( n = 10, weight = 1, df = 1, mean = 0, sd = 1 )

dmixt( x, weight = 1, df = 1, mean = 0, sd = 1 )
```

### Arguments

n	number of observations.
x	vector of quantiles.
weight	vector of probability weights, with length equal to number of components ( $k$ ). This is assumed to sum to 1; if not, it is normalized.
df	vector of degrees of freedom ( $> 0$ , maybe non-integer). $df = \text{Inf}$ is allowed.
mean	vector of means.
sd	vector of standard deviations.

### Details

Sampling from finite mixture of t-distribution, with density:

$$Pr(x|\underline{w}, \underline{df}, \underline{\mu}, \underline{\sigma}) = \sum_{i=1}^k w_i t_{df}(x|\mu_i, \sigma_i),$$

where

$$t_{df}(x|\mu, \sigma) = \frac{\Gamma(\frac{df+1}{2})}{\Gamma(\frac{df}{2})\sqrt{\pi df}\sigma} \left( 1 + \frac{1}{df} \left( \frac{x - \mu}{\sigma} \right)^2 \right)^{-\frac{df+1}{2}}.$$

### Value

Generated data as an vector with size  $n$ .



**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**References**

Mohammadi, A., Salehi-Rad, M. R., and Wit, E. C. (2013) Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683-700, doi: [10.1007/s0018001203233](https://doi.org/10.1007/s0018001203233)

Mohammadi, A., and Salehi-Rad, M. R. (2012) Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics-Simulation and Computation*, 41(3):419-435, doi: [10.1080/03610918.2011.588358](https://doi.org/10.1080/03610918.2011.588358)

**See Also**

[rt](#), [rmixnorm](#), [rmixgamma](#)

**Examples**

```
## Not run:
n      = 10000
weight = c( 0.3, 0.5, 0.2 )
df     = c( 4 , 4 , 4 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixt( n = n, weight = weight, df = df, mean = mean, sd = sd )

hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixt = dmixt( x, weight, df, mean, sd )

lines( x, densmixt, lwd = 2 )

## End(Not run)
```

---

plot.bmixgamma

*Plot function for S3 class "bmixgamma"*

---

**Description**

Visualizes the results for function [bmixgamma](#).

**Usage**

```
## S3 method for class 'bmixgamma'
plot( x, ... )
```

**Arguments**

`x` An object of S3 class "bmixgamma", from function [bmixgamma](#).  
`...` System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[bmixgamma](#)

**Examples**

```
## Not run:
# simulating data from mixture of gamma with two components
n      = 500 # number of observations
weight = c( 0.6, 0.4 )
alpha  = c( 12 , 1   )
beta   = c( 3  , 2   )

data <- rmixgamma( n = n, weight = weight, alpha = alpha, beta = beta )

# plot for simulation data
hist( data, prob = TRUE, nclass = 50, col = "gray" )

x      = seq( 0, 10, 0.05 )
truth  = dmixgamma( x, weight, alpha, beta )

lines( x, truth, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixgamma.obj <- bmixgamma( data )

plot( bmixgamma.obj )

## End(Not run)
```

---

plot.bmixnorm

*Plot function for S3 class "bmixnorm"*

---

**Description**

Visualizes the results for function [bmixnorm](#).

**Usage**

```
## S3 method for class 'bmixnorm'
plot( x, ... )
```

**Arguments**

x                    An object of S3 class "bmixnorm", from function [bmixnorm](#).  
 ...                   System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[bmixnorm](#)

**Examples**

```
## Not run:
# simulating data from mixture of Normal with 3 components
n      = 500
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixnorm.obj = bmixnorm( data, k = 3 )

plot( bmixnorm.obj )

## End(Not run)
```

---

plot.bmixt

*Plot function for S3 class "bmixt"*

---

**Description**

Visualizes the results for function [bmixt](#).

**Usage**

```
## S3 method for class 'bmixt'
plot( x, ... )
```

**Arguments**

x                    An object of S3 class "bmixt", from function [bmixt](#).  
 ...                  System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[bmixt](#)

**Examples**

```
## Not run:
# simulating data from mixture of Normal with 3 components
n      = 500
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixt.obj = bmixt( data, k = 3 )

plot( bmixt.obj )

## End(Not run)
```

---

```
print.bmixgamma            Print function for S3 class "bmixgamma"
```

---

**Description**

Prints the information about the output of function [bmixgamma](#).

**Usage**

```
## S3 method for class 'bmixgamma'
print( x, ... )
```

**Arguments**

x                    An object of S3 class "bmixgamma", from function [bmixgamma](#).  
...                   System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[bmixgamma](#)

**Examples**

```
## Not run:
# simulating data from mixture of gamma with two components
n      = 500 # number of observations
weight = c( 0.6, 0.4 )
alpha  = c( 12 , 1   )
beta   = c( 3  , 2   )

data <- rmixgamma( n = n, weight = weight, alpha = alpha, beta = beta )

# plot for simulation data
hist( data, prob = TRUE, nclass = 50, col = "gray" )

x      = seq( 0, 10, 0.05 )
truth  = dmixgamma( x, weight, alpha, beta )

lines( x, truth, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixgamma.obj <- bmixgamma( data, iter = 500 )

print( bmixgamma.obj )

## End(Not run)
```

---

print.bmixnorm                    *Print function for S3 class "bmixnorm"*

---

**Description**

Prints the information about the output of function [bmixnorm](#).

**Usage**

```
## S3 method for class 'bmixnorm'
print( x, ... )
```

**Arguments**

x                    An object of S3 class "bmixnorm", from function [bmixnorm](#).  
 ...                  System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[bmixnorm](#)

**Examples**

```
## Not run:
# simulating data from mixture of Normal with 3 components
n      = 500
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixnorm.obj = bmixnorm( data, k = 3, iter = 1000 )

print( bmixnorm.obj )

## End(Not run)
```

---

```
print.bmixt
```

```
Print function for S3 class "bmixt"
```

---

**Description**

Prints the information about the output of function [bmixt](#).

**Usage**

```
## S3 method for class 'bmixt'
print( x, ... )
```

**Arguments**

x                    object of S3 class "bmixt", from function [bmixt](#).  
...                   System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[bmixt](#)

**Examples**

```
## Not run:  
# simulating data from mixture of Normal with 3 components  
n        = 500  
weight = c( 0.3, 0.5, 0.2 )  
mean    = c( 0 , 10 , 3 )  
sd      = c( 1 , 1 , 1 )  
  
data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )  
  
# plot for simulation data  
hist( data, prob = TRUE, nclass = 30, col = "gray" )  
  
x            = seq( -20, 20, 0.05 )  
densmixnorm = dmixnorm( x, weight, mean, sd )  
  
lines( x, densmixnorm, lwd = 2 )  
  
# Running bdmcmc algorithm for the above simulation data set  
bmixt.obj = bmixt( data, k = 3, iter = 1000 )  
  
print( bmixt.obj )  
  
## End(Not run)
```

---

rdirichlet

*Random generation for the Dirichlet distribution*

---

**Description**

Random generation from the Dirichlet distribution.

**Usage**

```
rdirichlet( n = 10, alpha = c( 1, 1 ) )
```

**Arguments**

n                    number of observations.  
alpha                vector of shape parameters.

**Details**

The Dirichlet distribution is the multidimensional generalization of the beta distribution.

**Value**

A matrix with n rows, each containing a single Dirichlet random deviate.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**

[Beta](#)

**Examples**

```
draws = rdirichlet( n = 500, alpha = c( 1, 1, 1 ) )  
  
boxplot( draws )
```

---

summary.bmixgamma            *Summary function for S3 class "bmixgamma"*

---

**Description**

Provides a summary of the results for function [bmixgamma](#).

**Usage**

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

**Arguments**

object                An object of S3 class "bmixgamma", from function [bmixgamma](#).  
...                    System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>



**See Also**[bmixgamma](#)**Examples**

```
## Not run:
# simulating data from mixture of gamma with two components
n      = 500 # number of observations
weight = c( 0.6, 0.4 )
alpha  = c( 12 , 1   )
beta   = c( 3  , 2   )

data <- rmixgamma( n = n, weight = weight, alpha = alpha, beta = beta )

# plot for simulation data
hist( data, prob = TRUE, nclass = 50, col = "gray" )

x      = seq( 0, 10, 0.05 )
truth  = dmixgamma( x, weight, alpha, beta )

lines( x, truth, lwd = 2 )

# Running bdcmc algorithm for the above simulation data set
bmixgamma.obj <- bmixgamma( data, iter = 500 )

summary( bmixgamma.obj )

## End(Not run)
```

summary.bmixnorm

*Summary function for S3 class "bmixnorm"***Description**

Provides a summary of the results for function [bmixnorm](#).

**Usage**

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

**Arguments**

**object**            An object of S3 class "bmixnorm", from function [bmixnorm](#).  
**...**             System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**[bmixnorm](#)**Examples**

```
## Not run:
# simulating data from mixture of Normal with 3 components
n      = 500
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Running bdcmc algorithm for the above simulation data set
bmixnorm.obj = bmixnorm( data, k = 3, iter = 1000 )

summary( bmixnorm.obj )

## End(Not run)
```

---

`summary.bmixt`*Summary function for S3 class "bmixt"*

---

**Description**

Provides a summary of the results for function [bmixt](#).

**Usage**

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

**Arguments**

`object` An object of S3 class "bmixt", from function [bmixt](#).  
`...` System reserved (no specific usage).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**See Also**[bmixt](#)**Examples**

```
## Not run:
# simulating data from mixture of Normal with 3 components
n      = 500
weight = c( 0.3, 0.5, 0.2 )
mean   = c( 0 , 10 , 3 )
sd     = c( 1 , 1 , 1 )

data = rmixnorm( n = n, weight = weight, mean = mean, sd = sd )

# plot for simulation data
hist( data, prob = TRUE, nclass = 30, col = "gray" )

x      = seq( -20, 20, 0.05 )
densmixnorm = dmixnorm( x, weight, mean, sd )

lines( x, densmixnorm, lwd = 2 )

# Runing bdmcmc algorithm for the above simulation data set
bmixt.obj = bmixt( data, k = 3, iter = 1000 )

summary( bmixt.obj )

## End(Not run)
```

# Index

- \* **datasets**
    - galaxy, [12](#)
  - \* **distribution**
    - mixgamma, [13](#)
    - mixnorm, [14](#)
    - mixt, [16](#)
    - rdirichlet, [23](#)
  - \* **hplot**
    - plot.bmixgamma, [17](#)
    - plot.bmixnorm, [18](#)
    - plot.bmixt, [19](#)
  - \* **iteration**
    - bmixgamma, [3](#)
    - bmixnorm, [6](#)
    - bmixt, [9](#)
  - \* **package**
    - bmixture-package, [2](#)
  - \* **print**
    - print.bmixgamma, [20](#)
    - print.bmixnorm, [21](#)
    - print.bmixt, [22](#)
  - \* **sampling algorithms**
    - bmixgamma, [3](#)
    - bmixnorm, [6](#)
    - bmixt, [9](#)
- bmixnorm\_fixed\_k (bmixture-package), [2](#)  
bmixnorm\_unknown\_k  
(bmixture-package), [2](#)
- Beta, [24](#)  
bmixgamma, [3](#), [6](#), [8](#), [11](#), [17](#), [18](#), [20](#), [21](#), [24](#), [25](#)  
bmixnorm, [6](#), [6](#), [11](#), [18](#), [19](#), [21](#), [22](#), [25](#), [26](#)  
bmixt, [6](#), [8](#), [9](#), [19](#), [20](#), [22](#), [23](#), [26](#), [27](#)  
bmixture-package, [2](#)
- dmixgamma (mixgamma), [13](#)  
dmixnorm (mixnorm), [14](#)  
dmixt (mixt), [16](#)
- galaxy, [12](#)
- mixgamma, [13](#)  
mixnorm, [14](#)  
mixt, [16](#)
- plot.bmixgamma, [17](#)  
plot.bmixnorm, [18](#)  
plot.bmixt, [19](#)  
print.bmixgamma, [20](#)  
print.bmixnorm, [21](#)  
print.bmixt, [22](#)
- rdirichlet, [23](#)  
rgamma, [14](#)  
rmixgamma, [15](#), [17](#)  
rmixgamma (mixgamma), [13](#)  
rmixnorm, [8](#), [14](#), [17](#)  
rmixnorm (mixnorm), [14](#)  
rmixt, [11](#), [14](#), [15](#)  
rmixt (mixt), [16](#)  
rnorm, [15](#)  
rt, [17](#)
- summary.bmixgamma, [24](#)  
summary.bmixnorm, [25](#)  
summary.bmixt, [26](#)