# Package 'sanba'

May 23, 2025

**Type** Package

**Title** Fitting Shared Atoms Nested Models via MCMC or Variational Bayes

**Version** 0.0.1

**Date** 2025-05-15

**Maintainer** Francesco Denti <francescodenti.personal@gmail.com>

**URL** https://github.com/fradenti/sanba

**BugReports** https://github.com/fradenti/sanba/issues

**Description** An efficient tool for fitting nested mixture models based on a shared set of atoms via Markov Chain Monte Carlo and variational inference algorithms. Specifically, the package implements the common atoms model (Denti et al., 2023), its finite version (similar to D'Angelo et al., 2023), and a hybrid finite-infinite model (D'Angelo and Denti, 2024).
All models implement univariate nested mixtures with Gaussian kernels equipped with a normal-inverse gamma prior distribution on the parameters. Additional functions are provided to help analyze the results of the fitting procedure.
References:
Denti, Camerlenghi, Guindani, Mira (2023) <doi:10.1080/01621459.2021.1933499>,
D'Angelo, Canale, Yu, Guindani (2023) <doi:10.1111/biom.13626>,
D'Angelo, Denti (2024) <doi:10.1214/24-BA1458>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** Rcpp, matrixStats, salso

**Depends** scales, RColorBrewer

**LinkingTo** cpp11, Rcpp, RcppArmadillo, RcppProgress

**Language** en-US

**NeedsCompilation** yes

**Author** Francesco Denti [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-5034-7414>),
Laura D'Angelo [aut] (ORCID: <https://orcid.org/0000-0003-2978-4702>)

**Repository** CRAN

**Date/Publication** 2025-05-23 18:00:02 UTC

# Contents



---

`estimate_G` *Estimate the Atoms and Weights of the Discrete Mixing Distributions*

---

## Description

The function computes the posterior means of the atoms and weights characterizing the discrete mixing distributions. The function takes as input an object from `fit_CAM`, `fit_fiSAN`, or `fit_fSAN`, used with the `est_method = "VI"` argument, and returns an object of class `SANvi_G`.

## Usage

```
estimate_G(object, ...)

## S3 method for class 'SANvi_G'
plot(x, DC_num = NULL, lim = 2, ...)

## S3 method for class 'SANvi_G'
print(x, thr = 0.01, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `SANvi`. |
| `...` | ignored. |
| `x` | an object of class `SANvi_G` (usually, the result of a call to `estimate_G`). |
| `DC_num` | an integer or a vector of integers indicating which distributional clusters to plot. |
| `lim` | optional value for the `plot` method to adjust the limits of the x-axis (the default is 2). The atoms are plotted on a range given by `min(posterior means)-lim`, `max(posterior means)+lim`. |
| `thr` | argument for the `print` method. It should be a small positive number, representing a threshold. If the posterior weight of a specific shared atom is below the threshold, the atom is not reported. |

## Value

The function estimate_G returns an object of class SANvi_G, which is a matrix comprising the posterior means, variances, and weights of each estimated DC (one mixture component for each row).

## Examples

```
# Generate example data
set.seed(1232)
y <- c(rnorm(100),rnorm(100,5))
g <- rep(1:2,rep(100,2))

# Fitting fiSAN via variational inference
est <- fit_fiSAN(y,g)

# Estimate posterior atoms and weights
est <- estimate_G(est)
est
plot(est)
plot(est, DC_num = 1)
```

---

# fit_CAM *Fit the Common Atoms Mixture Model*

---

## Description

fit_CAM fits the common atoms mixture model (CAM) of Denti et al. (2023) with Gaussian kernels and normal-inverse gamma priors on the unknown means and variances. The function returns an object of class SANmcmc or SANvi depending on the chosen computational approach (MCMC or VI).

## Usage

```
fit_CAM(y, group, est_method = c("VI", "MCMC"),
         prior_param = list(),
         vi_param = list(),
         mcmc_param = list())
```

## Arguments

| | |
|---|---|
| y | Numerical vector of observations (required). |
| group | Numerical vector of the same length of y, indicating the group membership (required). |
| est_method | Character, specifying the preferred estimation method. It can be either "VI" or "MCMC". |
| prior_param | A list containing |

`m0, tau0, lambda0, gamma0` Hyperparameters on $(\mu, \sigma^2) \sim NIG(m_0, \tau_0, \lambda_0, \gamma_0)$. The default is (0, 0.01, 3, 2).

`hyp_alpha1, hyp_alpha2` If a random $\alpha$ is used, (`hyp_alpha1`, `hyp_alpha2`) specify the hyperparameters. The default is (1,1). The prior is $\alpha$ ~ Gamma(`hyp_alpha1`, `hyp_alpha2`).

`alpha` Distributional DP parameter if fixed (optional). The distribution is $\pi \sim GEM(\alpha)$.

`hyp_beta1, hyp_beta2` If a random $\beta$ is used, (`hyp_beta1`, `hyp_beta2`) specify the hyperparameters. The default is (1,1). The prior is $\beta$ ~ Gamma(`hyp_beta1`, `hyp_beta2`).

`beta` Observational DP parameter if fixed (optional). The distribution is $\omega_k \sim GEM(\beta)$.

`vi_param` A list of variational inference-specific settings containing

`maxL, maxK` Integers, the upper bounds for the observational and distributional clusters to fit, respectively. The default is (50, 20).

`epsilon` The tolerance that drives the convergence criterion adopted as stopping rule.

`seed` Random seed to control the initialization.

`maxSIM` The maximum number of CAVI iterations to perform.

`warmstart` Logical, if `TRUE`, the observational means of the cluster atoms are initialized with a k-means algorithm.

`verbose` Logical, if `TRUE` the iterations are printed.

`mcmc_param` A list of MCMC inference-specific settings containing

`nrep, burn` Integers, the number of total MCMC iterations, and the number of discarded iterations, respectively.

`maxL, maxK` Integers, the upper bounds for the observational and distributional clusters to fit, respectively. The default is (50, 20).

`seed` Random seed to control the initialization.

`warmstart` Logical, if `TRUE`, the observational means of the cluster atoms are initialized with a k-means algorithm. If `FALSE`, the starting points can be passed through the parameters `nclus_start, mu_start, sigma2_start, M_start, S_start, alpha_start, beta_start`

`verbose` Logical, if `TRUE` the iterations are printed.

## Details

The common atoms mixture model is used to perform inference in nested settings, where the data are organized into $J$ groups. The data should be continuous observations $(Y_1, \ldots, Y_J)$, where each $Y_j = (y_{1,j}, \ldots, y_{n_j,j})$ contains the $n_j$ observations from group $j$, for $j = 1, \ldots, J$. The function takes as input the data as a numeric vector `y` in this concatenated form. Hence, `y` should be a vector of length $n_1 + \cdots + n_J$. The `group` parameter is a numeric vector of the same size as `y`, indicating the group membership for each individual observation. Notice that with this specification, the observations in the same group need not be contiguous as long as the correspondence between the variables `y` and `group` is maintained.

### Model

The data are modeled using a Gaussian likelihood, where both the mean and the variance are observational cluster-specific, i.e.,

$$y_{i,j} \mid M_{i,j} = l \sim N(\mu_l, \sigma_l^2)$$

where $M_{i,j} \in \{1, 2, \dots\}$ is the observational cluster indicator of observation $i$ in group $j$. The prior on the model parameters is a normal-inverse gamma distribution $(\mu_l, \sigma_l^2) \sim NIG(m_0, \tau_0, \lambda_0, \gamma_0)$, i.e., $\mu_l \mid \sigma_l^2 \sim N(m_0, \sigma_l^2/\tau_0)$, $1/\sigma_l^2 \sim Gamma(\lambda_0, \gamma_0)$ (shape, rate).

**Clustering**

The model clusters both observations and groups. The clustering of groups (distributional clustering) is provided by the allocation variables $S_j \in \{1, 2, \dots\}$, with

$$Pr(S_j = k \mid \dots) = \pi_k \qquad \text{for } k = 1, 2, \dots$$

The distribution of the probabilities is $\{\pi_k\}_{k=1}^{\infty} \sim GEM(\alpha)$, where GEM is the Griffiths-Engen-McCloskey distribution of parameter $\alpha$, which characterizes the stick-breaking construction of the DP (Sethuraman, 1994).

The clustering of observations (observational clustering) is provided by the allocation variables $M_{i,j} \in \{1, 2, \dots\}$, with

$$Pr(M_{i,j} = l \mid S_j = k, \dots) = \omega_{l,k} \qquad \text{for } k = 1, 2, \dots ;\ l = 1, 2, \dots$$

The distribution of the probabilities is $\{\omega_{l,k}\}_{l=1}^{\infty} \sim GEM(\beta)$ for all $k = 1, 2, \dots$

## Value

`fit_CAM` returns a list of class `SANvi`, if `method = "VI"`, or `SANmcmc`, if `method = "MCMC"`. The list contains the following elements:

`model` Name of the fitted model.

`params` List containing the data and the parameters used in the simulation. Details below.

`sim` List containing the optimized variational parameters or the simulated values. Details below.

`time` Total computation time.

**Data and parameters**: `params` is a list with the following components:

- `y, group, Nj, J`: Data, group labels, group frequencies, and number of groups.
- `K, L`: Number of distributional and observational mixture components.
- `m0, tau0, lambda0, gamma0`: Model hyperparameters.
- (`hyp_alpha1, hyp_alpha2`) or `alpha`: hyperparameters on $\alpha$ (if $\alpha$ random); or provided value for $\alpha$ (if fixed).
- (`hyp_beta1, hyp_beta2`) or `beta`: hyperparameters on $\beta$ (if $\beta$ random); or provided value for $\beta$ (if fixed).
- `seed`: The random seed adopted to replicate the run.
- `epsilon, n_runs`: If `method = "VI"`, the threshold controlling the convergence criterion and the number of iterations needed to reach convergence.
- `nrep, burnin`: If `method = "MCMC"`, the number of total MCMC iterations, and the number of discarded ones.

**Simulated values**: depending on the algorithm, it returns a list with the optimized variational parameters or a list with the chains of the simulated values.

**Variational inference**: `sim` is a list with the following components:

- `theta_l`: Matrix of size (`maxL`, 4). Each row is a posterior variational estimate of the four normal-inverse gamma hyperparameters.
- `XI`: A list of length J. Each element is a matrix of size (N, `maxL`) containing the posterior variational probability of assignment of the i-th observation in the j-th group to the l-th OC, i.e., $\hat{\xi}_{i,j,l} = \hat{\mathbb{Q}}(M_{i,j} = l)$.
- `RHO`: Matrix of size (J, `maxK`). Each row is a posterior variational probability of assignment of the j-th group to the k-th DC, i.e., $\hat{\rho}_{j,k} = \hat{\mathbb{Q}}(S_j = k)$.
- `a_tilde_k, b_tilde_k`: Vector of updated variational parameters of the beta distributions governing the distributional stick-breaking process.
- `a_bar_lk, b_bar_lk`: Matrix of updated variational parameters of the beta distributions governing the observational stick-breaking process (arranged by column).
- `conc_hyper`: If the concentration parameters are chosen to be random, a vector with the four updated hyperparameters.
- `Elbo_val`: Vector containing the values of the ELBO.

**MCMC inference**: `sim` is a list with the following components:

- `mu`: Matrix of size (`nrep`, `maxL`). Each row is a posterior sample of the mean parameter for each observational cluster $(\mu_1, \dots \mu_L)$.
- `sigma2`: Matrix of size (`nrep`, `maxL`). Each row is a posterior sample of the variance parameter for each observational cluster $(\sigma_1^2, \dots \sigma_L^2)$.
- `obs_cluster`: Matrix of size (`nrep`, n), with n = `length(y)`. Each row is a posterior sample of the observational cluster allocation variables $(M_{1,1}, \dots, M_{n_J,J})$.
- `distr_cluster`: Matrix of size (`nrep`, J), with J = `length(unique(group))` Each row is a posterior sample of the distributional cluster allocation variables $(S_1, \dots, S_J)$.
- `pi`: Matrix of size (`nrep`, `maxK`). Each row is a posterior sample of the distributional cluster probabilities $(\pi_1, \dots, \pi_{maxK})$.
- `omega`: 3-d array of size (`maxL`, `maxK`, `nrep`). Each slice is a posterior sample of the observational cluster probabilities. In each slice, each column $k$ is a vector (of length `maxL`) observational cluster probabilities $(\omega_{1,k}, \dots, \omega_{maxL,k})$ for distributional cluster $k$.
- `alpha`: Vector of length `nrep` of posterior samples of the parameter $\alpha$.
- `beta`: Vector of length `nrep` of posterior samples of the parameter $\beta$.
- `maxK`: Vector of length `nrep` of the number of distributional DP components used by the slice sampler.
- `maxL`: Vector of length `nrep` of the number of observational DP components used by the slice sampler.

## References

Denti, F., Camerlenghi, F., Guindani, M., and Mira, A. (2023). A Common Atoms Model for the Bayesian Nonparametric Analysis of Nested Data. *Journal of the American Statistical Association*, 118(541), 405-416. DOI: 10.1080/01621459.2021.1933499

Sethuraman, A.J. (1994). A Constructive Definition of Dirichlet Priors, *Statistica Sinica*, 4, 639–650.

## Examples

```
set.seed(123)
y <- c(rnorm(60), rnorm(40, 5))
g <- rep(1:2, rep(50, 2))
plot(density(y[g==1]), xlim = c(-5,10), main = "Group-specific density")
lines(density(y[g==2]), col = 2)

out_vi <- fit_CAM(y, group = g, est_method = "VI", vi_param = list(n_runs = 1))
out_vi

out_mcmc <- fit_CAM(y = y, group = g, est_method = "MCMC",
                    mcmc_param = list(nrep = 500, burn = 100))
out_mcmc
```

---

`fit_fiSAN` *Fit the Finite-Infinite Shared Atoms Mixture Model*

---

## Description

`fit_fiSAN` fits the finite-infinite shared atoms nested (fiSAN) mixture model with Gaussian kernels and normal-inverse gamma priors on the unknown means and variances. The function returns an object of class `SANmcmc` or `SANvi` depending on the chosen computational approach (MCMC or VI).

## Usage

```
fit_fiSAN(y, group, est_method = c("VI", "MCMC"),
          prior_param = list(),
          vi_param = list(),
          mcmc_param = list())
```

## Arguments

| | |
|---|---|
| `y` | Numerical vector of observations (required). |
| `group` | Numerical vector of the same length of `y`, indicating the group membership (required). |
| `est_method` | Character, specifying the preferred estimation method. It can be either `"VI"` or `"MCMC"`. |
| `prior_param` | A list containing: |
| | `m0, tau0, lambda0, gamma0` Hyperparameters on $(\mu, \sigma^2) \sim NIG(m_0, \tau_0, \lambda_0, \gamma_0)$. The default is (0, 0.01, 3, 2). |
| | `hyp_alpha1, hyp_alpha2` If a random $\alpha$ is used, (`hyp_alpha1`, `hyp_alpha2`) specify the hyperparameters. The default is (1,1). The prior is $\alpha$ ~ Gamma(`hyp_alpha1`, `hyp_alpha2`). |
| | `alpha` Distributional DP parameter if fixed (optional). The distribution is $\pi \sim$ GEM($\alpha$). |

b_dirichlet The hyperparameter of the symmetric observational Dirichlet distribution. The default is 1/maxL.

vi_param A list of variational inference-specific settings containing:

maxL, maxK Integers, the upper bounds for the observational and distributional clusters to fit, respectively. The default is (50, 20).

epsilon The tolerance that drives the convergence criterion adopted as stopping rule.

seed Random seed to control the initialization.

maxSIM The maximum number of CAVI iterations to perform.

warmstart Logical, if TRUE, the observational means of the cluster atoms are initialized with a k-means algorithm.

verbose Logical, if TRUE the iterations are printed.

mcmc_param A list of MCMC inference-specific settings containing:

nrep, burn Integers, the number of total MCMC iterations, and the number of discarded iterations, respectively.

maxL, maxK Integers, the upper bounds for the observational and distributional clusters to fit, respectively. The default is (50, 20).

seed Random seed to control the initialization.

warmstart Logical, if TRUE, the observational means of the cluster atoms are initialized with a k-means algorithm. If FALSE, the starting points can be passed through the parameters nclus_start, mu_start, sigma2_start, M_start, S_start, alpha_start.

verbose Logical, if TRUE the iterations are printed.

## Details

### Data structure

The finite-infinite common atoms mixture model is used to perform inference in nested settings, where the data are organized into $J$ groups. The data should be continuous observations $(Y_1, \ldots, Y_J)$, where each $Y_j = (y_{1,j}, \ldots, y_{n_j,j})$ contains the $n_j$ observations from group $j$, for $j = 1, \ldots, J$. The function takes as input the data as a numeric vector y in this concatenated form. Hence, y should be a vector of length $n_1 + \cdots + n_J$. The group parameter is a numeric vector of the same size as y, indicating the group membership for each individual observation. Notice that with this specification, the observations in the same group need not be contiguous as long as the correspondence between the variables y and group is maintained.

### Model

The data are modeled using a Gaussian likelihood, where both the mean and the variance are observational-cluster-specific:

$$y_{i,j} \mid M_{i,j} = l \sim N(\mu_l, \sigma_l^2)$$

where $M_{i,j} \in \{1, \ldots, L\}$ is the observational cluster indicator of observation $i$ in group $j$. The prior on the model parameters is a normal-inverse gamma distribution $(\mu_l, \sigma_l^2) \sim NIG(m_0, \tau_0, \lambda_0, \gamma_0)$, i.e., $\mu_l \mid \sigma_l^2 \sim N(m_0, \sigma_l^2/\tau_0)$, $1/\sigma_l^2 \sim$ Gamma$(\lambda_0, \gamma_0)$ (shape, rate).

### Clustering

The model clusters both observations and groups. The clustering of groups (distributional clustering) is provided by the allocation variables $S_j \in \{1, 2, \dots\}$, with:

$$Pr(S_j = k \mid \dots) = \pi_k \qquad \text{for } k = 1, 2, \dots$$

The distribution of the probabilities is $\{\pi_k\}_{k=1}^{\infty} \sim GEM(\alpha)$, where GEM is the Griffiths-Engen-McCloskey distribution of parameter $\alpha$, which characterizes the stick-breaking construction of the DP (Sethuraman, 1994).

The clustering of observations (observational clustering) is provided by the allocation variables $M_{i,j} \in \{1, \dots, L\}$, with:

$$Pr(M_{i,j} = l \mid S_j = k, \dots) = \omega_{l,k} \qquad \text{for } k = 1, 2, \dots ;\ l = 1, \dots, L.$$

The distribution of the probabilities is $(\omega_{1,k}, \dots, \omega_{L,k}) \sim \text{Dirichlet}_L(b, \dots, b)$ for all $k = 1, 2, \dots$. Here, the dimension $L$ is fixed.

## Value

`fit_fiSAN` returns a list of class `SANvi`, if `method = "VI"`, or `SANmcmc`, if `method = "MCMC"`. The list contains the following elements:

`model` Name of the fitted model.

`params` List containing the data and the parameters used in the simulation. Details below.

`sim` List containing the optimized variational parameters or the simulated values. Details below.

`time` Total computation time.

**Data and parameters**: `params` is a list with the following components:

- `y, group, Nj, J`: Data, group labels, group frequencies, and number of groups.
- `K, L`: Number of distributional and observational mixture components.
- `m0, tau0, lambda0, gamma0`: Model hyperparameters.
- (`hyp_alpha1, hyp_alpha2`) or `alpha`: Hyperparameters on $\alpha$ (if $\alpha$ random); or provided value for $\alpha$ (if fixed).
- `b_dirichlet`: Provided value for $b$.
- `seed`: The random seed adopted to replicate the run.
- `epsilon, n_runs`: If `method = "VI"`, the threshold controlling the convergence criterion and the number of iterations needed to reach convergence.
- `nrep, burnin`: If `method = "MCMC"`, the number of total MCMC iterations, and the number of discarded ones.

**Simulated values**: Depending on the algorithm, it returns a list with the optimized variational parameters or a list with the chains of the simulated values.

**Variational inference**: `sim` is a list with the following components:

- `theta_l`: Matrix of size (`maxL`, 4). Each row is a posterior variational estimate of the four normal-inverse gamma hyperparameters.
- `XI`: A list of length J. Each element is a matrix of size (N, `maxL`), the posterior variational assignment probabilities $\hat{\mathbb{Q}}(M_{i,j} = l)$.

- RHO: Matrix of size (J, maxK), with the posterior variational assignment probabilities $\hat{\mathbb{Q}}(S_j = k)$.
- a_tilde_k, b_tilde_k: Vector of updated variational parameters of the beta distributions governing the distributional stick-breaking process.
- conc_hyper: If the concentration parameter is random, this contains its updated hyperparameters.
- b_dirichlet_lk: Matrix of updated variational parameters of the Dirichlet distributions governing observational clustering.
- Elbo_val: Vector containing the values of the ELBO.

**MCMC inference**: sim is a list with the following components:

- mu: Matrix of size (nrep, maxL) with samples of the observational cluster means.
- sigma2: Matrix of size (nrep, maxL) with samples of the observational cluster variances.
- obs_cluster: Matrix of size (nrep, n) with posterior samples of the observational cluster allocations.
- distr_cluster: Matrix of size (nrep, J) with posterior samples of the distributional cluster allocations.
- pi: Matrix of size (nrep, maxK) with posterior samples of the distributional cluster weights.
- omega: Array of size (maxL, maxK, nrep) with observational cluster weights.
- alpha: Vector of length nrep with posterior samples of $\alpha$.
- maxK: Vector of length nrep with number of active distributional components.

## Examples

```
set.seed(123)
y <- c(rnorm(60), rnorm(40, 5))
g <- rep(1:2, rep(50, 2))
plot(density(y[g==1]), xlim = c(-5,10), main = "Group-specific density")
lines(density(y[g==2]), col = 2)

out_vi <- fit_fiSAN(y, group = g, est_method = "VI",
                    vi_param = list(n_runs = 1))
out_vi

out_mcmc <- fit_fiSAN(y = y, group = g, est_method = "MCMC")
out_mcmc
```

---

fit_fSAN *Fit the Finite Shared Atoms Mixture Model*

---

## Description

fit_fSAN fits the finite shared atoms nested (fSAN) mixture model with Gaussian kernels and normal-inverse gamma priors on the unknown means and variances. The function returns an object of class SANmcmc or SANvi depending on the chosen computational approach (MCMC or VI).

## Usage

```
fit_fSAN(y, group, est_method = c("VI", "MCMC"),
         prior_param = list(),
         vi_param = list(),
         mcmc_param = list())
```

## Arguments

| | |
|---|---|
| y | Numerical vector of observations (required). |
| group | Numerical vector of the same length of y, indicating the group membership (required). |
| est_method | Character, specifying the preferred estimation method. It can be either "VI" or "MCMC". |
| prior_param | A list containing |
| | m0, tau0, lambda0, gamma0 Hyperparameters on $(\mu, \sigma^2) \sim NIG(m_0, \tau_0, \lambda_0, \gamma_0)$. The default is (0, 0.01, 3, 2). |
| | a_dirichlet The hyperparameter of the symmetric distributional Dirichlet distribution. The default is 1/maxK. |
| | b_dirichlet The hyperparameter of the symmetric observational Dirichlet distribution. The default is 1/maxL. |
| vi_param | A list of variational inference-specific settings, containing |
| | maxL, maxK Integers, the upper bounds for the observational and distributional clusters to fit, respectively. The default is (50, 20). |
| | epsilon The tolerance that drives the convergence criterion adopted as stopping rule. |
| | seed Random seed to control the initialization. |
| | maxSIM The maximum number of CAVI iteration to perform. |
| | warmstart Logical, if TRUE, the observational means of the cluster atoms are initialized with a k-means algorithm. |
| | verbose Logical, if TRUE the iterations are printed. |
| mcmc_param | A list of MCMC inference-specific settings, containing |
| | nrep, burn Integers, the number of total MCMC iterations, and the number of discarded iterations, respectively. |
| | maxL, maxK Integers, the upper bounds for the observational and distributional clusters to fit, respectively. The default is (50, 20). |
| | seed Random seed to control the initialization. |
| | warmstart Logical, if TRUE, the observational means of the cluster atoms are initialized with a k-means algorithm. If FALSE, the starting points can be passed through the parameters nclus_start, mu_start, sigma2_start, M_start, S_start |
| | verbose Logical, if TRUE the iterations are printed. |

## Details

### Data structure

The finite common atoms mixture model is used to perform inference in nested settings, where the data are organized into $J$ groups. The data should be continuous observations $(Y_1, \dots, Y_J)$, where each $Y_j = (y_{1,j}, \dots, y_{n_j,j})$ contains the $n_j$ observations from group $j$, for $j = 1, \dots, J$. The function takes as input the data as a numeric vector `y` in this concatenated form. Hence `y` should be a vector of length $n_1 + \cdots + n_J$. The `group` parameter is a numeric vector of the same size as `y` indicating the group membership for each individual observation. Notice that with this specification the observations in the same group need not be contiguous as long as the correspondence between the variables `y` and `group` is maintained.

### Model

The data are modeled using a Gaussian likelihood, where both the mean and the variance are observational-cluster-specific, i.e.,

$$y_{i,j} \mid M_{i,j} = l \sim N(\mu_l, \sigma_l^2)$$

where $M_{i,j} \in \{1, \dots, L\}$ is the observational cluster indicator of observation $i$ in group $j$. The prior on the model parameters is a normal-inverse gamma distribution $(\mu_l, \sigma_l^2) \sim NIG(m_0, \tau_0, \lambda_0, \gamma_0)$, i.e., $\mu_l \mid \sigma_l^2 \sim N(m_0, \sigma_l^2/\tau_0)$, $1/\sigma_l^2 \sim Gamma(\lambda_0, \gamma_0)$ (shape, rate).

### Clustering

The model performs a clustering of both observations and groups. The clustering of groups (distributional clustering) is provided by the allocation variables $S_j \in \{1, \dots, K\}$, with

$$Pr(S_j = k \mid \dots) = \pi_k \qquad \text{for } k = 1, \dots, K.$$

The distribution of the probabilities is $(\pi_1, \dots, \pi_K) \sim Dirichlet_K(a, \dots, a)$. Here, the dimension $K$ is fixed.

The clustering of observations (observational clustering) is provided by the allocation variables $M_{i,j} \in \{1, \dots, L\}$, with

$$Pr(M_{i,j} = l \mid S_j = k, \dots) = \omega_{l,k} \qquad \text{for } k = 1, \dots, K\,;\ l = 1, \dots, L.$$

The distribution of the probabilities is $(\omega_{1,k}, \dots, \omega_{L,k}) \sim Dirichlet_L(b, \dots, b)$ for all $k = 1, \dots, K$. Here, the dimension $L$ is fixed.

## Value

`fit_fSAN` returns a list of class `SANvi`, if `method = "VI"`, or `SANmcmc`, if `method = "MCMC"`. The list contains the following elements:

`model` Name of the fitted model.

`params` List containing the data and the parameters used in the simulation. Details below.

`sim` List containing the optimized variational parameters or the simulated values. Details below.

`time` Total computation time.

**Data and parameters**: `params` is a list with the following components:

- `y, group, Nj, J`: Data, group labels, group frequencies, and number of groups.

- `K, L`: Number of distributional and observational mixture components.
- `m0, tau0, lambda0, gamma0`: Model hyperparameters.
- `a_dirichlet`: Provided value for $a$.
- `b_dirichlet`: Provided value for $b$.
- `seed`: The random seed adopted to replicate the run.
- `epsilon, n_runs`: If `method = "VI"`, the threshold controlling the convergence criterion and the number of iterations needed to reach convergence.
- `nrep, burnin`: If `method = "MCMC"`, the number of total MCMC iterations, and the number of discarded ones.

**Simulated values**: depending on the algorithm, it returns a list with the optimized variational parameters or a list with the chains of the simulated values.

**Variational inference**: `sim` is a list with the following components:

- `theta_l`: Matrix of size (`maxL`, 4). Each row is a posterior variational estimate of the four normal-inverse gamma hyperparameters.
- `XI` : A list of length J. Each element is a matrix of size (N, `maxL`) posterior variational probability of assignment of assignment of the i-th observation in the j-th group to the l-th OC, i.e., $\hat{\xi}_{i,j,l} = \hat{\mathbb{Q}}(M_{i,j} = l)$.
- `RHO`: Matrix of size (J, `maxK`). Each row is a posterior variational probability of assignment of the j-th group to the k-th DC, i.e., $\hat{\rho}_{j,k} = \hat{\mathbb{Q}}(S_j = k)$.
- `a_dirichlet_k`: Vector of updated variational parameters of the Dirichlet distribution governing the distributional clustering.
- `b_dirichlet_lk`: Matrix of updated variational parameters of the Dirichlet distributions governing the observational clustering (arranged by column).
- `Elbo_val`: Vector containing the values of the ELBO.

**MCMC inference**: `sim` is a list with the following components:

- `mu`: Matrix of size (`nrep`, `maxL`). Each row is a posterior sample of the mean parameter of each observational cluster $(\mu_1, \dots \mu_L)$.
- `sigma2`: Matrix of size (`nrep`, `maxL`). Each row is a posterior sample of the variance parameter of each observational cluster $(\sigma_1^2, \dots \sigma_L^2)$.
- `obs_cluster`: Matrix of size (`nrep`, n), with n = `length(y)`. Each row is a posterior sample of the observational cluster allocation variables $(M_{1,1}, \dots, M_{n_J,J})$.
- `distr_cluster`: Matrix of size (`nrep`, J), with J = `length(unique(group))`. Each row is a posterior sample of the distributional cluster allocation variables $(S_1, \dots, S_J)$.
- `pi`: Matrix of size (`nrep`, `maxK`). Each row is a posterior sample of the distributional cluster probabilities $(\pi_1, \dots, \pi_{\texttt{maxK}})$.
- `omega`: 3-d array of size (`maxL`, `maxK`, `nrep`). Each slice is a posterior sample of the observational cluster probabilities. In each slice, each column $k$ is a vector (of length `maxL`) observational cluster probabilities $(\omega_{1,k}, \dots, \omega_{\texttt{maxL},k})$ for distributional cluster $k$.

## Examples

```
set.seed(123)
y <- c(rnorm(60), rnorm(40, 5))
g <- rep(1:2, rep(50, 2))
plot(density(y[g==1]), xlim = c(-5,10), main = "Group-specific density")
lines(density(y[g==2]), col = 2)

out_vi <- fit_fSAN(y, group = g, est_method = "VI", vi_param = list(n_runs = 1))
out_vi

out_mcmc <- fit_fSAN(y = y, group = g, est_method = "MCMC",
                     mcmc_param = list(nrep = 100, burn= 50))
out_mcmc
```

---

`plot.SANmcmc` *Visual check of convergence of the MCMC output*

---

## Description

Plot method for objects of class `SANmcmc`. Check the convergence of the MCMC through visual inspection of the chains.

## Usage

```
## S3 method for class 'SANmcmc'
plot(
  x,
  param = c("mu", "sigma2", "pi", "num_clust", "alpha", "beta"),
  show_density = TRUE,
  add_burnin = 0,
  show_convergence = TRUE,
  trunc_plot = 2,
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | Object of class `SANmcmc` (usually, the result of a call to `fit_CAM`, `fit_fiSAN`, or `fit_fSAN`, used with the `est_method = "MCMC"` argument). |
| `param` | String with the names of the parameters to check. It can be one of `"mu"`, `"sigma2"`, `"pi"`, `"num_clust"`, `"alpha"`, `"beta"`. |
| `show_density` | Logical (default `TRUE`). Should a kernel estimate of the density be plotted? |
| `add_burnin` | Integer (default = 0). Additional number of observations to discard in the burn-in. |

show_convergence
: Logical (default TRUE). Should a superimposed red line of the cumulative mean be plotted?

trunc_plot
: Integer (default = 10). For multidimensional parameters, the maximum number of components to be plotted.

...
: Ignored.

## Value

The function displays the traceplots and posterior density estimates of the parameters sampled in the MCMC algorithm.

## Note

The function is not available for the observational weights $\omega$.

## Examples

```
set.seed(123)
y <- c(rnorm(40,0,0.3), rnorm(20,5,0.3))
g <- c(rep(1,30), rep(2, 30))
out <- fit_fiSAN(y = y, group = g, "MCMC", mcmc_param = list(nrep = 500, burn = 200))
plot(out, param = "mu", trunc_plot = 2)
plot(out, param = "sigma2", trunc_plot = 2)
plot(out, param = "alpha", trunc_plot = 1)
plot(out, param = "alpha", add_burnin = 100)
plot(out, param = "pi", trunc_plot = 4, show_density = FALSE)

out <- fit_CAM(y = y, group = g, "MCMC",
mcmc_param = list(nrep = 500, burn = 200, seed= 1234))
plot(out, param = "mu", trunc_plot = 2)
plot(out, param = "sigma2", trunc_plot = 2)
plot(out, param = "alpha")
plot(out, param = "pi", trunc_plot = 2)
plot(out, param = "pi", trunc_plot = 5)
plot(out, param = "num_clust", trunc_plot = 5)
plot(out, param = "beta", trunc_plot = 2)

out <- fit_fSAN(y = y, group = g, "MCMC", mcmc_param = list(nrep = 500, burn = 200))
plot(out, param = "mu", trunc_plot = 2)
plot(out, param = "sigma2", trunc_plot = 2)
plot(out, param = "pi", trunc_plot = 4,
     show_convergence = FALSE, show_density = FALSE)
```

## plot.SANvi *Visual check of convergence of the VI output*

### Description

Plot method for objects of class SANvi. The function displays two graphs. The left plot shows the progression of all the ELBO values as a function of the iterations. The right plots shows the ELBO increments between successive iterations of the best run on a log scale (note: increments should always be positive).

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class SANvi (usually, the result of a call to fit_CAM, fit_fiSAN, or fit_fSAN, used with the est_method = "VI" argument). |
| ... | Ignored. |

### Value

The function plots the path followed by the ELBO and its subsequent differences.

### Examples

```
set.seed(123)
y <- c(rnorm(200,0,0.3), rnorm(100,5,0.3))
g <- c(rep(1,150), rep(2, 150))
out <- fit_fSAN(y = y, group = g, "VI", vi_param = list(n_runs = 2))
plot(out)
```

## print.SANmcmc *Print the MCMC output*

### Description

Print method for objects of class SANmcmc.

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class SANmcmc. |
| ... | Ignored. |

### Value

The function prints a summary of the fitted model.

---

## print.SANvi *Print the variational inference output*

---

### Description

Print method for objects of class SANvi.

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class SANvi. |
| ... | Further arguments passed to or from other methods. |

### Value

The function prints a summary of the fitted model.

---

## summary *Summarize the estimated observational and distributional partition*

---

### Description

Given the output of a sanba model-fitting function, estimate the observational and distributional partitions using salso::salso() for MCMC, and the maximum a posteriori estimate for VI.

## Usage

```
## S3 method for class 'SANvi'
summary(object, ordered = TRUE, ...)

## S3 method for class 'SANmcmc'
summary(object, ordered = TRUE, add_burnin = 0, ncores = 0, ...)

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

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

## S3 method for class 'summary_mcmc'
plot(
  x,
  DC_num = NULL,
  type = c("ecdf", "boxplot", "scatter"),
  alt_palette = FALSE,
  ...
)

## S3 method for class 'summary_vi'
plot(
  x,
  DC_num = NULL,
  type = c("ecdf", "boxplot", "scatter"),
  alt_palette = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| object | Object of class SANmcmc (usually, the result of a call to fit_fiSAN, fit_fSAN, or fit_CAM with method = "MCMC") or SANvi (the result of a call to fit_fiSAN,fit_fSAN, or fit_CAM with method = "VI"). |
| ordered | Logical, if TRUE (default), the function sorts the distributional cluster labels reflecting the increasing values of medians of the data assigned to each DC. |
| ... | Additional graphical parameters to be passed to the plot function. |
| add_burnin | Integer (default = 0). Number of observations to discard as additional burn-in (only for SANmcmc objects). |
| ncores | A parameter to pass to the salso::salso() function (only for SANmcmc objects). The number of CPU cores to use for parallel computing; a value of zero indicates the use of all cores on the system. |
| x | The result of a call to summary. |
| DC_num | An integer or a vector of integers indicating which distributional clusters to plot. |

| | |
|---|---|
| type | What type of plot should be drawn. Available types are "boxplot", "ecdf", and "scatter". |
| alt_palette | Logical, the color palette to be used. Default is R base colors (alt_palette = FALSE). |

## Value

A list of class summary_vi or summary_mcmc containing

- obs_level: a data frame containing the data values, their group indexes, and the observational and distributional clustering assignments for each observation.
- dis_level: a vector with the distributional clustering assignment for each unit.

## See Also

salso::salso(), print.SANmcmc, plot.SANmcmc

## Examples

```
set.seed(123)
y <- c(rnorm(40,0,0.3), rnorm(20,5,0.3))
g <- c(rep(1:6, each = 10))
out <- fit_fSAN(y = y, group = g, "VI", vi_param = list(n_runs = 10))
plot(out)
clust <- summary(out)
clust
plot(clust, lwd = 2, alt_palette = TRUE)
plot(clust, type = "scatter", alt_palette = FALSE, cex = 2)


set.seed(123)
y <- c(rnorm(40,0,0.3), rnorm(20,5,0.3))
g <- c(rep(1:6, each = 10))
out <- fit_fSAN(y = y, group = g, "MCMC", mcmc_param=list(nrep=500,burn=200))
plot(out)
clust <- summary(out)
clust
plot(clust, lwd = 2)
plot(clust,  type = "boxplot", alt_palette = TRUE)
plot(clust,  type = "scatter", alt_palette = TRUE, cex = 2, pch = 4)
```

# Index