

Package: SSLfmm (via r-universe)

May 8, 2026

Type Package

Title Semi-Supervised Learning under a Mixed-Missingness Mechanism in Finite Mixture Models

Version 0.1.0

Maintainer Jinran Wu <jinran.wu@uq.edu.au>

Description Implements a semi-supervised learning framework for finite mixture models under a mixed-missingness mechanism. The approach models both missing completely at random (MCAR) and entropy-based missing at random (MAR) processes using a logistic-entropy formulation. Estimation is carried out via an Expectation-Conditional Maximisation (ECM) algorithm with robust initialisation routines for stable convergence. The methodology relates to the statistical perspective and informative missingness behaviour discussed in Ahfok and McLachlan (2020) <doi:10.1007/s11222-020-09971-5> and Ahfok and McLachlan (2023) <doi:10.1016/j.ecosta.2022.03.007>. The package provides functions for data simulation, model estimation, prediction, and theoretical Bayes error evaluation for analysing partially labelled data under a mixed-missingness mechanism.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports stats, mvtnorm, matrixStats

NeedsCompilation no

Author Jinran Wu [aut, cre] (ORCID: <<https://orcid.org/0000-0002-2388-3614>>), Geoffrey J. McLachlan [aut] (ORCID: <<https://orcid.org/0000-0002-5921-3145>>)

Repository <https://cran.r-universe.dev>

Date/Publication 2025-12-09 16:30:28 UTC

RemoteUrl <https://github.com/cran/SSLfmm>

RemoteRef HEAD

RemoteSha 37c3c6a407052c30b28a769833fc81e8382bc763

Contents

bayesclassifier	2
compute_d2	3
EM_FMM_SemiSupervised	4
EM_FMM_SemiSupervised_Complete_Initial	6
EM_FMM_SemiSupervised_Initial	7
error_beta_classification	8
get_clusterprobs	9
get_entropy	11
initialestimate	12
logsumexp	13
neg_loglik	13
normalise_logprob	15
pack_theta	15
rmix	16
simulate_mixed_missingness	17
unpack_theta	18

Index **19**

bayesclassifier	<i>Bayes' Rule Classifier</i>
-----------------	-------------------------------

Description

Classifier specified by Bayes' rule. Assigns $\arg \max_i \{\log(\pi_i) + \log \mathcal{N}(y_j | \mu_i, \Sigma_i)\}$.

Usage

```
bayesclassifier(dat, p, g, pi = NULL, mu = NULL, sigma = NULL, paralist = NULL)
```

Arguments

<code>dat</code>	An $n \times p$ numeric matrix (or a length- p numeric vector, treated as $1 \times p$).
<code>p</code>	Integer; dimension of the observation vector.
<code>g</code>	Integer; number of Gaussian components/classes.
<code>pi</code>	Numeric length- g vector of mixing proportions (must sum to 1).
<code>mu</code>	Either: <ul style="list-style-type: none"> • A $p \times g$ numeric matrix (column i is μ_i), or • A length-g list of length-p numeric vectors.
<code>sigma</code>	Either:

- A $p \times p$ covariance matrix (shared), or
 - A length- g list of $p \times p$ covariance matrices, or
 - A $p \times p \times g$ numeric array.
- paralist Optional list with elements pi, mu, sigma (overrides explicit args if provided and non-NULL).

Value

An integer vector of length n with predicted class labels in $1 : g$.

Examples

```
# Minimal example with list-style mu and sigma:
set.seed(1)
p <- 2; g <- 2
pi <- c(0.5, 0.5)
mu <- list(`1`=c(0.96, 0.02), `2`=c(-1.02, -0.03))
sigma <- list(
  matrix(c(0.9417379,0.5447264, 0.5447264,0.9811853), 2, 2, byrow=TRUE),
  matrix(c(0.9984812,0.3314474, 0.3314474,1.1316865), 2, 2, byrow=TRUE)
)
X <- mvtnorm::rmvnorm(5, mean = mu[[1]], sigma = sigma[[1]])
bayesclassifier(X, p=p, g=g, pi=pi, mu=mu, sigma=sigma)
```

compute_d2

Squared Discriminant Score for Two-Group LDA (Equal Covariance)

Description

Computes d^2 where $d = \beta_0 + y^\top \beta_1$ for two classes under the common-covariance (LDA) model:

$$\beta_1 = \Sigma^{-1}(\mu_1 - \mu_2), \quad \beta_0 = \log(\pi_1/\pi_2) - \frac{1}{2}(\mu_1 + \mu_2)^\top \Sigma^{-1}(\mu_1 - \mu_2).$$

Usage

```
compute_d2(y, mu1, mu2, Sigma_inv, pi1, pi2)
```

Arguments

- y Numeric vector (length p) or numeric matrix with p columns; rows are observations.
- mu1, mu2 Numeric vectors of length p : class means.
- Sigma_inv $p \times p$ numeric precision matrix (inverse covariance).
- pi1, pi2 Positive scalars: class prior probabilities (need not sum to 1).

Value

If y is a vector, a single numeric d^2 . If y is a matrix, a numeric vector of d^2 values for each row.

Examples

```
set.seed(1)
mu1 <- c(0, 0); mu2 <- c(1, 1)
S <- matrix(c(1, .2, .2, 1), 2, 2)
Sigma_inv <- solve(S)
x <- c(0.5, -0.2)
compute_d2(x, mu1, mu2, Sigma_inv, pi1 = 0.6, pi2 = 0.4)

X <- matrix(rnorm(10 * 2), ncol = 2)
compute_d2(X, mu1, mu2, Sigma_inv, 0.5, 0.5)
```

EM_FMM_SemiSupervised *EM for Semi-Supervised FMM with a Mixed-Missingness Mechanism (MCAR + entropy-based MAR)*

Description

Runs an EM-like procedure that models a mixed-missingness mechanism: unlabeled indicator m_j follows a mixed-missingness mechanism with MCAR (prob α) and entropy-based MAR via a logistic link $q_j = \text{logit}^{-1}(\xi_0 + \xi_1 \log e_j)$. Supports shared (ncov = 1) or class-specific (ncov = 2) covariance.

Usage

```
EM_FMM_SemiSupervised(
  data,
  g = 2,
  init_res,
  max_iter = 5,
  tol = 1e-06,
  ncov = 1,
  verbose = FALSE
)
```

Arguments

<code>data</code>	A data.frame or matrix with $p+2$ columns: first p are features, then missing (0 = labelled, 1 = unlabelled), and z (class label for labelled rows; ignored otherwise).
<code>g</code>	Integer, number of mixture components (classes).
<code>init_res</code>	A list with initial parameters: <ul style="list-style-type: none"> • <code>pi</code>: numeric length-g (mixture weights, sum to 1) • <code>mu</code>: list of length g, each length-p mean vector

	<ul style="list-style-type: none"> • Sigma: if $ncov = 1$, a $p \times p$ matrix; if $ncov = 2$, a list of $g \times p \times p$ matrices • alpha: scalar in $(0,1)$ • xi: numeric length-2, logistic coefficients (ξ_0, ξ_1)
max_iter	Integer, max EM iterations.
tol	Convergence tolerance on log-likelihood increase.
ncov	Integer covariance structure: 1 = shared/equal, 2 = class-specific/unequal.
verbose	Logical; if TRUE, progress messages are printed using message(). Default is FALSE.

Details

This function expects the following helpers to be available:

- pack_theta($\pi_k, \mu_k, \Sigma_k, g, p, ncov$)
- unpack_theta($\theta, g, p, ncov$)
- neg_loglik($\theta, Y_{all}, m_j, Z_{all}, d2_{yj}, \xi, \alpha_k, unpacker$)
- get_entropy($dat, n, p, g, paralist$) returning per-observation entropy-like values

Value

A list with elements: $\pi, \mu, \Sigma, \xi, \alpha, \text{loglik}$, and $ncov$.

Examples

```
## Toy example using a simple partially-labelled dataset
set.seed(1)

## 1) Construct an n x (p+2) partially-labelled dataset:
##   first p columns = features, then 'missing' and 'z'
n <- 100; p <- 2; g <- 2

X <- matrix(rnorm(n * p), nrow = n, ncol = p)

## missing: 0 = labelled, 1 = unlabelled
missing <- rbinom(n, size = 1, prob = 0.3)

## z: observed class labels for labelled rows, NA for unlabelled
z <- rep(NA_integer_, n)
z[missing == 0] <- sample(1:g, sum(missing == 0), replace = TRUE)

sim_dat <- data.frame(X, missing = missing, z = z)

## 2) Warm-up initialisation using the complete-data initializer
init <- EM_FMM_SemiSupervised_Complete_Initial(
  data = sim_dat,
  g = g,
  ncov = 1
)
```

```

## 3) Run the main EM algorithm (small number of iterations)
fit <- EM_FMM_SemiSupervised(
  data      = sim_dat,
  g         = g,
  init_res  = init,
  ncov      = 1,
  max_iter  = 5,
  verbose   = FALSE
)

str(fit)

```

EM_FMM_SemiSupervised_Complete_Initial

*Complete-Data Warm-Up Initialization for Semi-Supervised FMM
with a Mixed-Missingness Mechanism*

Description

Uses both labeled and unlabeled subsets of the data to obtain quick initial estimates for mixture parameters and missingness mechanism parameters (α , ξ) via a warm-up EM procedure.

Usage

```

EM_FMM_SemiSupervised_Complete_Initial(
  data,
  g = 2,
  ncov = 1,
  alpha_init = 0.01,
  warm_up_iter = 200,
  tol = 1e-06
)

```

Arguments

<code>data</code>	A data frame containing: <ul style="list-style-type: none"> • The first p columns: numeric variables used in the FMM. • A column <code>missing</code>: indicator (0 = labeled, 1 = unlabeled/missing). • A column <code>z</code>: class labels for labeled rows (1 : g); NA for unlabeled.
<code>g</code>	Integer, number of mixture components (default 2).
<code>ncov</code>	Integer, covariance structure: 1 = shared (equal), 2 = class-specific (unequal).
<code>alpha_init</code>	Numeric in (0,1), initial MCAR proportion (default 0.01).
<code>warm_up_iter</code>	Integer, number of warm-up EM iterations (default 200).
<code>tol</code>	Convergence tolerance on α (default 1e-6).

Details

- This function first calls `initialestimate` to get initial π, μ, Σ .
- Then it calls `EM_FMM_SemiSupervised_Initial` with these values for a short warm-up run.
- Covariance structure (equal vs. unequal) is determined by `ncov`.

Value

A list with initial values for `EM_FMM_SemiSupervised`:

- `pi` - mixture weights.
- `mu` - list of component mean vectors.
- `Sigma` - covariance matrix/matrices.
- `alpha` - MCAR proportion.
- `xi` - logistic regression coefficients for MAR mechanism.

`EM_FMM_SemiSupervised_Initial`

Quick Initializer for alpha, xi, and Mixture Parameters

Description

Provides rough initial estimates of the missingness parameters `alpha` and `xi`, together with mixture parameters `pi`, `mu`, and covariance matrices, using a lightweight EM-style routine. The covariance structure is chosen automatically based on `Sigma_init`:

- If `Sigma_init` is a $p \times p$ matrix, a **shared (equal)** covariance is used.
- If `Sigma_init` is a list of length `g` of $p \times p$ matrices or a $p \times p \times g$ array, **class-specific (unequal)** covariances are used.
- If `Sigma_init` is NULL, a shared covariance is estimated from the labeled data.

This function is intended as a fast, heuristic initializer rather than a final estimator for the mixed missingness model.

Usage

```
EM_FMM_SemiSupervised_Initial(
  Y_labelled,
  Z_labelled,
  Y_unlabelled,
  g = 2,
  pi_init = NULL,
  mu_init = NULL,
  Sigma_init = NULL,
  alpha_init = 0.01,
  warm_up_iter = 50,
  tol = 1e-06
)
```

Arguments

<code>Y_labelled</code>	Numeric matrix of labeled observations ($n_L \times p$).
<code>Z_labelled</code>	Integer vector of class labels in $1:g$ for <code>Y_labelled</code> .
<code>Y_unlabelled</code>	Numeric matrix of unlabeled observations ($n_U \times p$).
<code>g</code>	Integer, number of mixture components (default 2).
<code>pi_init</code>	Optional numeric length- g vector of initial mixing proportions.
<code>mu_init</code>	Optional list of length g of initial mean vectors (each of length p).
<code>Sigma_init</code>	Optional initial covariance: a $p \times p$ matrix (shared), or a list of g $p \times p$ matrices, or a $p \times p \times g$ array (class-specific).
<code>alpha_init</code>	Numeric in $(0, 1)$, initial MCAR proportion (default 0.01).
<code>warm_up_iter</code>	Integer, number of warm-up EM iterations used to refine the quick initial estimates (default 50).
<code>tol</code>	Convergence tolerance on <code>alpha</code> (default $1e-6$).

Value

A list with elements:

- `pi` - length- g vector of mixing proportions.
- `mu` - list of g mean vectors.
- `Sigma` - shared $p \times p$ matrix (equal-Sigma) or list of g matrices (unequal-Sigma).
- `xi` - length-2 numeric vector $c(x_{i0}, x_{i1})$ from the logistic MAR model.
- `alpha` - estimated MCAR proportion.
- `gamma` - $n \times g$ responsibility matrix.
- `d2_yj` - numeric vector of entropy-based scores used in the missingness model.

Note

This is a heuristic warm-up routine. It requires helper functions `normalise_logprob()` and `get_entropy()` to be available in the package namespace.

`error_beta_classification`

Compute Theoretical Bayes' Error for a Binary Gaussian Mixture

Description

Computes the Bayes' classification error rate for a two-component Gaussian mixture given `mu_hat`, `Sigma_hat`, and `pi_hat`. If `mu_hat` is supplied as a list of length 2, it is converted to a $p \times 2$ matrix internally.

Usage

```
error_beta_classification(mu_hat, Sigma_hat, pi_hat)
```

Arguments

mu_hat Either a numeric matrix of size $p \times 2$ whose columns are component means, or a list of two numeric vectors.

Sigma_hat Numeric $p \times p$ covariance matrix shared across components.

pi_hat Numeric vector of length 2 with mixing proportions (π_1, π_2) that are non-negative and sum to 1.

Details

The linear discriminant is

$$\beta_1 = \Sigma^{-1}(\mu_1 - \mu_2), \quad \beta_0 = -\frac{1}{2}(\mu_1 + \mu_2)^\top \Sigma^{-1}(\mu_1 - \mu_2) + \log(\pi_1/\pi_2)$$

and the Bayes' error is

$$\text{Err} = \sum_{k=1}^2 \pi_k \Phi \left(\frac{(-1)^k \{\beta_0 + \beta_1^\top \mu_k\}}{\|\beta_1\|} \right),$$

where Φ is the standard normal cdf.

Value

A numeric scalar giving the theoretical Bayes' classification error rate.

Examples

```
mu_hat <- matrix(c(1, 0, -1, 0), nrow = 2) # columns are mu1, mu2
Sigma_hat <- diag(2)
pi_hat <- c(0.5, 0.5)
error_beta_classification(mu_hat, Sigma_hat, pi_hat)
```

```
get_clusterprobs
```

Posterior Cluster Probabilities for a Gaussian Mixture

Description

Compute posterior membership probabilities τ_{ij} for each observation under a Gaussian mixture with either a shared covariance matrix or component-specific covariances.

Usage

```
get_clusterprobs(
  dat,
  n,
  p,
  g,
  pi = NULL,
  mu = NULL,
  sigma = NULL,
  paralist = NULL
)
```

Arguments

<code>dat</code>	Numeric matrix $n \times p$. Data matrix (rows = observations).
<code>n</code>	Integer. Number of observations (checked against <code>nrow(dat)</code>).
<code>p</code>	Integer. Number of variables (checked against <code>ncol(dat)</code>).
<code>g</code>	Integer. Number of components.
<code>pi</code>	Optional numeric vector length g . Mixing proportions (sum to 1).
<code>mu</code>	Optional numeric matrix $p \times g$. Column i is mean of component i .
<code>sigma</code>	Optional numeric matrix $p \times p$ (shared covariance) or array $p \times p \times g$ (component-specific covariances).
<code>paralist</code>	Optional list with elements <code>pi</code> , <code>mu</code> , <code>sigma</code> . If provided, these take precedence over the corresponding explicit args.

Value

Numeric matrix $n \times g$ of posterior probabilities τ_{ij} .

Examples

```
n <- 100; p <- 2; g <- 2
X <- matrix(rnorm(n*p), n, p)
pi <- c(0.6, 0.4)
mu <- cbind(c(0,0), c(1,1))
Sig <- array(0, dim = c(p,p,g)); Sig[,,1] <- diag(p); Sig[,,2] <- diag(p)
tau <- get_clusterprobs(X, n, p, g, pi = pi, mu = mu, sigma = Sig)
head(tau)
```

get_entropy

Per-Row Entropy of Posterior Cluster Probabilities

Description

Compute the Shannon entropy (in nats) of posterior membership probabilities for each observation, given a Gaussian mixture model. Posterior probabilities τ_{ij} are obtained via `get_clusterprobs()`.

Usage

```
get_entropy(dat, n, p, g, pi = NULL, mu = NULL, sigma = NULL, paralist = NULL)
```

Arguments

<code>dat</code>	Numeric matrix $n \times p$. The data matrix.
<code>n</code>	Integer. Number of rows in <code>dat</code> .
<code>p</code>	Integer. Number of columns (features) in <code>dat</code> .
<code>g</code>	Integer. Number of mixture components.
<code>pi</code>	Optional numeric vector length g . Mixing proportions (sum to 1).
<code>mu</code>	Optional numeric matrix $p \times g$. Column j is the mean of component j .
<code>sigma</code>	Optional numeric matrix $p \times p$ (shared covariance) or array $p \times p \times g$ (component-specific covariances).
<code>paralist</code>	Optional list with elements <code>pi</code> , <code>mu</code> , <code>sigma</code> . If supplied, these take precedence over the corresponding explicit arguments.

Value

Numeric vector of length n , the entropy per observation (in nats).

See Also

[get_clusterprobs](#)

Examples

```
# Suppose you have get_clusterprobs(), and a fitted/pi, mu, sigma:
n <- 100; p <- 2; g <- 2
X <- matrix(rnorm(n * p), n, p)
pi <- c(0.6, 0.4)
mu <- cbind(c(0,0), c(1,1))
Sigma <- array(0, dim = c(p, p, g))
Sigma[, ,1] <- diag(p); Sigma[, ,2] <- diag(p)
ent <- get_entropy(dat = X, n = n, p = p, g = g, pi = pi, mu = mu, sigma = Sigma)
summary(ent)
```

initialestimate *Initialize Parameters for a FMM from Labeled Subset*

Description

Builds initial estimates (π, μ, Σ) for a g -component Gaussian mixture using only rows with observed labels in zm . Supports either a shared covariance ($ncov = 1$) or class-specific covariances ($ncov = 2$).

Usage

```
initialestimate(dat, zm, g, ncov = 2, ridge = 1e-06)
```

Arguments

<code>dat</code>	A numeric matrix or data frame of features ($n \times p$).
<code>zm</code>	Integer vector of length n with class labels in $1:g$; use NA for unlabeled rows. Only labeled rows contribute to the initialization.
<code>g</code>	Integer, number of mixture components.
<code>ncov</code>	Integer, 1 for a shared covariance matrix, 2 for class-specific covariance matrices. Default 2.
<code>ridge</code>	Numeric, small diagonal ridge added to covariance(s) for numerical stability. Default $1e-6$.

Details

If a class has zero or one labeled sample, its covariance is set to the global empirical covariance (from labeled data) with a small ridge. Class means for empty classes default to the global mean with a small jitter.

Value

A list with

- `pi`: length- g vector of mixing proportions (summing to 1).
- `mu`: $p \times g$ matrix of class means (column i is μ_i).
- `sigma`: if $ncov = 1$, a $p \times p$ shared covariance matrix; if $ncov = 2$, a $p \times p \times g$ array of class-specific covariances.

Examples

```
set.seed(1)
n <- 50; p <- 3; g <- 2
X <- matrix(rnorm(n*p), n, p)
z <- sample(c(1:g, NA), n, replace = TRUE, prob = c(0.4, 0.4, 0.2))
init <- initialestimate(X, z, g, ncov = 2)
str(init)
```

logsumexp	<i>Numerically Stable Log-Sum-Exp</i>
-----------	---------------------------------------

Description

Computes $\log(\sum_i \exp(x_i))$ in a numerically stable way by subtracting the maximum value before exponentiation.

Usage

```
logsumexp(x)
```

Arguments

x A numeric vector.

Value

A single numeric value: the log-sum-exp of x.

Examples

```
logsumexp(c(1000, 1001, 1002))
```

neg_loglik	<i>Negative Log-Likelihood for Semi-Supervised FMM with a Mixed-Missingness Mechanism</i>
------------	-------------------------------------------------------------------------------------------

Description

Computes the negative log-likelihood for a semi-supervised Gaussian mixture model under a mixed missingness mechanism (MCAR + entropy-based MAR). Assumes a covariance matrix Σ across all mixture components.

Usage

```
neg_loglik(theta, Y, m_j, Z, d2_yj, xi, alpha_k, unpack_fn)
```

Arguments

theta Numeric vector of packed model parameters to be unpacked by unpack_fn.
Y Numeric matrix of observations (n x p).
m_j Integer or logical vector of length n indicating missingness: 0 for observed (labeled block), 1 for unlabeled/missingness block.

Z	Integer vector of length n with class labels for labeled samples (1..g); use NA for unlabeled rows.
d2_yj	Numeric vector of length n with the entropy-like score used in the MAR mechanism (e.g., posterior entropy or any scalar proxy).
xi	Numeric length-2 vector c(xi0, xi1) for the logistic MAR model $q_j = \text{logit}^{-1}(\xi_0 + \xi_1 d2_{yj})$.
alpha_k	Numeric scalar in (0,1), the MCAR mixing proportion in the missingness mechanism.
unpack_fn	Function that takes theta and returns a list with elements: pi Numeric vector of length g with mixture weights. mu List of length g; each element is a numeric mean vector (length p). sigma Shared covariance matrix (p x p).

Details

The total log-likelihood is composed of three parts:

1. Labeled samples ($m_j = 0$) with observed class labels Z_j .
2. Unlabeled samples attributed to MCAR with probability mass m_{1j} .
3. Unlabeled samples attributed to MAR with probability mass m_{2j} .

The MAR probability for each sample is $q_j = \text{logit}^{-1}(\xi_0 + \xi_1 d2_{yj})$. Internally, the function uses a numerically stable logSumExp.

Value

A single numeric value: the negative log-likelihood.

Note

This implementation is for the equal covariance case (shared Σ).

Examples

```
# Minimal example (illustrative only):
library(mvtnorm)
set.seed(1)
n <- 20; p <- 2; g <- 2
Y <- matrix(rnorm(n*p), n, p)
Z <- sample(c(1:g, rep(NA, n - g)), n, replace = TRUE)
m_j <- ifelse(is.na(Z), 1L, 0L)
d2_yj <- runif(n)
xi <- c(-1, 2)
alpha_k <- 0.4
unpack_fn <- function(theta) {
  list(pi = c(0.6, 0.4),
       mu = list(c(0,0), c(1,1)),
       sigma = diag(p))
}
```

```
theta <- numeric(1) # not used in this toy unpack_fn  
neg_loglik(theta, Y, m_j, Z, d2_yj, xi, alpha_k, unpack_fn)
```

normalise_logprob *Normalise Log-Probabilities*

Description

Converts log-probabilities into a probability distribution by exponentiating in a numerically stable way.

Usage

```
normalise_logprob(log_probs)
```

Arguments

log_probs A numeric vector of log-probabilities.

Value

A numeric vector summing to 1 (the normalised probabilities).

Examples

```
lp <- c(-1000, -999, -998)  
normalise_logprob(lp)
```

pack_theta *Pack FMM Parameters into a Vector*

Description

Packs mixture weights, means, and covariance(s) into a single numeric vector. Uses the last component as the baseline for mixture weights (g-1 logits stored).

Usage

```
pack_theta(pi_k, mu_k, Sigma, g, p, ncov = 1)
```

Arguments

pi_k	Numeric vector of length g with mixture weights (positive, sum to 1).
mu_k	List of length g ; each element a numeric vector of length p (component means).
Sigma	Covariance: if $ncov = 1$, a single $p \times p$ matrix; if $ncov = 2$, a list of g $p \times p$ matrices.
g	Integer: number of components.
p	Integer: dimension.
ncov	Integer: covariance structure; 1 for shared covariance, 2 for class-specific.

Value

Numeric vector with parameters packed.

rmix	<i>Draw from a Gaussian Mixture Model</i>
------	-------------------------------------------

Description

Generate i.i.d. samples from a finite Gaussian mixture with either a shared covariance matrix or component-specific covariance matrices.

Usage

```
rmix(n, pi, mu, sigma, seed_number)
```

Arguments

n	Integer. Number of observations to generate.
pi	Numeric vector of length g . Mixing proportions (must sum to 1).
mu	Numeric matrix $p \times g$. Column j is the mean for component j .
sigma	Either a numeric matrix $p \times p$ (shared covariance), or a numeric array $p \times p \times g$ (component-specific covariances).
seed_number	Integer. Seed for reproducibility.

Value

A list with:

Y	Numeric matrix $n \times p$ of generated features.
Z	Numeric matrix $n \times g$ of one-hot component indicators.
clust	Integer vector n , the component labels in $1:g$.

Examples

```

set.seed(1)
g <- 2; p <- 2
pi <- c(0.5, 0.5)
mu <- cbind(c(1,0), c(-1,0))
Sigma <- diag(p)
out <- rmix(500, pi, mu, Sigma, seed_number = 123)
str(out)

```

```
simulate_mixed_missingness
```

Simulate a Gaussian Mixture Dataset with a Mixed-Missingness Mechanism (MAR + MCAR)

Description

Simulate a Gaussian Mixture Dataset with a Mixed-Missingness Mechanism (MAR + MCAR)

Usage

```

simulate_mixed_missingness(
  n = 500,
  pi,
  mu,
  sigma,
  xi0 = 2,
  xi1 = 3,
  alpha = 0.1,
  seed_id = 123
)

```

Arguments

n	Integer; sample size.
pi	Numeric vector; mixing proportions (sum to 1).
mu	Matrix (p x K); component means, columns = components.
sigma	Array (p x p x K); component covariance matrices.
xi0	Numeric; MAR logit intercept.
xi1	Numeric; MAR logit slope on entropy.
alpha	Numeric in $[\theta, 1]$; MCAR rate applied within both MAR and observed groups.
seed_id	Integer; seed passed to rmix() (your generator).

Details

Requires user-provided functions:

- `rmix(n, pi, mu, sigma, seed_number)`
- `get_entropy(dat, n, p, g, paralist)`

Missingness mechanism codes:

- 0 = fully observed
- 1 = MCAR
- 2 = MAR (entropy-based)

Value

A list with:

- `data`: data.frame with columns `x1..xp`, `en`, `missing`, `label`, `truth`
- `true_setup`: list(pi, mu, sigma)
- `groups`: list(mar_group, obs_group, mcar_in_mar, mcar_in_obs)
- `probs`: vector `prob_mar`
- `raw`: original `rmix` output `dat` augmented with `en` and labels

unpack_theta

Unpack FMM Parameter Vector

Description

Unpacks mixture weights, means, and covariance(s) from a parameter vector.

Usage

```
unpack_theta(theta, g, p, ncov = 1)
```

Arguments

<code>theta</code>	Numeric vector as returned by <code>pack_theta()</code> .
<code>g</code>	Integer: number of components.
<code>p</code>	Integer: dimension.
<code>ncov</code>	Integer: covariance structure; 1 for shared covariance, 2 for class-specific.

Value

A list with:

- **pi** Mixture weights (length `g`).
- **mu** List of `g` mean vectors.
- **sigma** Shared covariance matrix (`ncov=1`) or list of `g` covariance matrices (`ncov=2`).

Index

bayesclassifier, 2

compute_d2, 3

EM_FMM_SemiSupervised, 4, 7

EM_FMM_SemiSupervised_Complete_Initial,
6

EM_FMM_SemiSupervised_Initial, 7, 7

error_beta_classification, 8

get_clusterprobs, 9, 11

get_entropy, 11

initialestimate, 7, 12

logsumexp, 13

neg_loglik, 13

normalise_logprob, 15

pack_theta, 15

rmix, 16

simulate_mixed_missingness, 17

unpack_theta, 18