Package ‘dmbc’

September 24, 2024

Type Package

Title Model Based Clustering of Binary Dissimilarity Measurements

Version 1.0.3

Date 2024-09-24

Description Functions for fitting a Bayesian model for grouping binary dissimilarity matrices in homogeneous clusters. Currently, it includes methods only for binary data (<doi:10.18637/jss.v100.i16>).

Author Sergio Venturini [aut, cre],
Raffaella Piccarreta [ctb]

Maintainer Sergio Venturini <`sergio.venturini@unicatt.it`>

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

LazyData TRUE

Imports abind, bayesplot (>= 1.7.0), coda (>= 0.19-3), ggplot2 (>= 3.2.1), ggrepel (>= 0.8.1), graphics, modeltools (>= 0.2-22), parallel (>= 3.6.1), robustbase (>= 0.93-5), robustX (>= 1.2-5), stats4 (>= 3.6.0), tools

Suggests knitr, mcmcplots, testthat

Depends methods, R (>= 3.6.0), stats, utils

LinkingTo Rcpp, RcppArmadillo, RcppProgress

BugReports `https://github.com/sergioventurini/dmbc/issues`

Encoding UTF-8

RoxygenNote 7.3.2

Date/Publication 2024-09-24 09:40:02 UTC

Contents

`adjust_x` *Adjustment of the center and orientation of a latent configuration.*

Description

`adjust_x` adjusts the center and orientation of a latent configuration in Bayesian (metric) multidimensional scaling (BMDS).

Usage

```
adjust_x(x)
```

Arguments

`x`	Numeric matrix containing the latent configuration.

Value

A list with elements:

`x` A real matrix containing the adjusted latent configuration.

`Sig_x` The variance and covariance matrix of the adjusted latent configuration.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

See Also

`bmds` for (one-way) Bayesian (metric) multidimensional scaling.

Examples

```
n <- 100
nr <- 20
nc <- floor(n/nr)
x <- matrix(rnorm(1:n), nrow = nr, ncol = nc)
adj_x <- adjust_x(x)
adj_x$x
adj_x$Sig_x
```

`animals` *List of binary dissimilarity matrices among 18 animals.*

Description

To illustrate the MDS analysis of sorting data, Takane et al. (2009) refer to judgments on the similarity between $n = 18$ animals expressed by $S = 20$ subjects. Each subject was asked to divide the animals into as many groups as needed, based on their similarity. We converted these values to 0 or 1 depending on whether a pair of animals is placed or not in the same group by a subject.

Usage

```
data(animals)
```

Format

A `dmbc_data` object whose `diss` element is a list of 20 binary dissimilarity matrices. Each matrix is defined as a `dist` object measuring whether each pair of the 18 animals has is placed in the same group (1) or not (0).

The `dist` objects have rows and columns that are named as follows:

be	bear
cm	camel
ct	cat
cw	cow
dg	dog
el	elephant
gf	giraffe
fx	fox
hs	horse
li	lion
mk	monkey
ms	mouse
pg	pig

rb rabbit

sh sheep

sq squirrel

tg tiger

wf wolf

References

Takane, Y., Jung, S., Takane, Y. O. (2009). "Multidimensional Scaling". In Millsap, R. E., Maydeu-Olivares, A. (eds.), The SAGE Handbook of Quantitative Methods in Psychology, chapter 10, pp. 217–242,.

Examples

```
data(animals)
library(bayesplot)
cols <- color_scheme_set("teal")
plot(animals, colors = unlist(cols)[c(1, 6)], font = 1, cex.font = 0.75)
```

`bmds` *Bayesian multidimensional scaling (BMDS) using Markov Chain Monte Carlo (MCMC).*

Description

bmds computes the Bayesian multidimensional scaling (BMDS) solutions using Markov Chain Monte Carlo for a range of specified latent space dimensions.

Usage

```
bmds(
  D,
  min_p = 1,
  max_pm1 = 6,
  burnin = 0,
  nsim = 13000,
  ic = TRUE,
  verbose = TRUE
)
```

Arguments

`D`	Observed dissimilarities (provided as a distance matrix).
`min_p`	A length-one numeric vector providing the minimum value of the latent space dimension to use.

`max_pm1`	A length-one numeric vector providing the maximum value of the latent space dimension to use (minus 1).
`burnin`	A length-one numeric vector providing the number of iterations to use for burnin.
`nsim`	A length-one numeric vector providing the number of iterations to use in the MCMC simulation after burnin.
`ic`	Logical scalar. If `TRUE` computes the MDS information criterion (MDSIC) for all solution requested.
`verbose`	Logical scalar. If `TRUE` prints information regarding the evolution of the simulation.

Value

A list with the following elements:

`x.chain` MCMC chain of the latent configuration coordinates.

`sigma.chain` MCMC chain of the random error.

`lambda.chain` MCMC chain of the latent configuration variances.

`stress` Numeric vector of the stress function values.

`mdsIC` List with two elements, the MDSIC and BIC values for the required solutions.

`accept` Numeric matrix of acceptance rates.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Oh, M.-S., Raftery, A. E. (2001), "Bayesian Multidimensional Scaling and Choice of Dimension", Journal of the American Statistical Association, 96, 1031-1044.

See Also

`cmdscale` for classical (metric) multidimensional scaling.

Examples

```
## Not run:
# Airline Distances Between Cities
airline <- read.csv(file = system.file("extdata", "airline.csv",
  package = "dmbc"))
airline.nm <- airline[, 1]
airline <- airline[, 2:31]
colnames(airline) <- airline.nm
airline <- as.dist(airline)

min_p <- 1
max_p <- 4
burnin <- 200
```

```
nsim <- 1000
totiter <- burnin + nsim

airline.mds <- cmdscale(airline, max_p)
airline.bmds <- bmds(airline, min_p, max_p, burnin, nsim)

opar <- par(mfrow = c(1, 2))
plot(min_p:max_p, airline.bmds$mdsIC$mdsic, type = "b",
  main = "MDS Information Criterion", xlab = "p", ylab = "MDSIC")
MDSICmin <- which.min(airline.bmds$mdsIC$mdsic)
points((min_p:max_p)[MDSICmin], airline.bmds$mdsIC$mdsic[MDSICmin],
  col = "red", pch = 10, cex = 1.75, lwd = 1.5)

airline.bmds.x.mode <- bmds_get_x_mode(airline, airline.bmds, MDSICmin,
  min_p, max_p, start = (burnin + 1), end = totiter)
airline.bmds.d <- dist(airline.bmds.x.mode)
airline.mds.d <- dist(airline.mds[, 1:((min_p:max_p)[MDSICmin])])
plot(airline, airline.bmds.d, type = "n", xlab = "observed",
  ylab = "estimated", main = "Airline Distances \n Between Cities",
  xlim = c(0, max(airline, airline.bmds.d)),
  ylim = c(0, max(airline, airline.bmds.d)))
abline(0, 1, lty = 2, col = "gray")
points(airline, airline.mds.d, pch = 19, col = "cyan", cex = .5)
points(airline, airline.bmds.d, pch = 19, col = "magenta", cex = .5)
legend(x = "bottomright", legend = c("Classical MDS", "Bayesian MDS"),
  pch = c(19, 19), col = c("cyan", "magenta"))
par(opar)

# Careers of Lloyds Bank Employees
lloyds <- read.csv(file = system.file("extdata", "lloyds.csv",
  package = "dmbc"))
lloyds.nm <- lloyds[, 1]
lloyds <- lloyds[, 2:81]
colnames(lloyds) <- lloyds.nm
lloyds <- as.dist(lloyds)

min_p <- 1
max_p <- 12
burnin <- 200
nsim <- 1000
totiter <- burnin + nsim

lloyds.mds <- cmdscale(lloyds, max_p)
lloyds.bmds <- bmds(lloyds, min_p, max_p, burnin, nsim)

opar <- par(mfrow = c(1, 2))
plot((min_p:max_p), lloyds.bmds$mdsIC$mdsic, type = "b",
  main = "MDS Information Criterion", xlab = "p", ylab = "MDSIC")
MDSICmin <- which.min(lloyds.bmds$mdsIC$mdsic)
points((min_p:max_p)[MDSICmin], lloyds.bmds$mdsIC$mdsic[MDSICmin],
  col = "red", pch = 10, cex = 1.75, lwd = 1.5)

lloyds.bmds.x.mode <- bmds_get_x_mode(lloyds, lloyds.bmds, MDSICmin,
```

```
  min_p, max_p, start = (burnin + 1), end = totiter)
lloyds.bmds.d <- dist(lloyds.bmds.x.mode)
lloyds.mds.d <- dist(lloyds.mds[, 1:((min_p:max_p)[MDSICmin])])
plot(lloyds, lloyds.bmds.d, type = "n", xlab = "observed",
  ylab = "estimated", main = "Careers of Lloyds \n Bank Employees, 1905-1950",
  xlim = c(0, max(lloyds, lloyds.bmds.d)),
  ylim = c(0, max(lloyds, lloyds.bmds.d)))
abline(0, 1, lty = 2, col = "gray")
points(lloyds, lloyds.mds.d, pch = 19, col = "cyan", cex = .5)
points(lloyds, lloyds.bmds.d, pch = 19, col = "magenta", cex = .5)
legend(x = "topleft", legend = c("Classical MDS", "Bayesian MDS"),
  pch = c(19, 19), col = c("cyan", "magenta"))
par(opar)

# Road distances (in km) between 21 cities in Europe
data(eurodist, package = "datasets")

min_p <- 1
max_p <- 10
burnin <- 200
nsim <- 1000
totiter <- burnin + nsim

eurodist.mds <- cmdscale(eurodist, max_p)
eurodist.bmds <- bmds(eurodist, min_p, max_p, burnin, nsim)

opar <- par(mfrow = c(1, 2))
plot((min_p:max_p), eurodist.bmds$mdsIC$mdsic, type = "b",
  main = "MDS Information Criterion", xlab = "p", ylab = "MDSIC")
MDSICmin <- which.min(eurodist.bmds$mdsIC$mdsic)
points((min_p:max_p)[MDSICmin], eurodist.bmds$mdsIC$mdsic[MDSICmin],
  col = "red", pch = 10, cex = 1.75, lwd = 1.5)

eurodist.bmds.x.mode <- bmds_get_x_mode(eurodist, eurodist.bmds,
  MDSICmin, min_p, max_p, start = (burnin + 1), end = totiter)
eurodist.bmds.d <- dist(eurodist.bmds.x.mode)
eurodist.mds.d <- dist(eurodist.mds[, 1:((min_p:max_p)[MDSICmin])])
plot(eurodist, eurodist.bmds.d, type = "n", xlab = "observed",
  ylab = "estimated", main = "Road distances (in km) \n between 21 cities in Europe",
  xlim = c(0, max(eurodist, eurodist.bmds.d)),
  ylim = c(0, max(eurodist, eurodist.bmds.d)))
abline(0, 1, lty = 2, col = "gray")
points(eurodist, eurodist.mds.d, pch = 19, col = "cyan", cex = .5)
points(eurodist, eurodist.bmds.d, pch = 19, col = "magenta", cex = .5)
legend(x = "topleft", legend = c("Classical MDS", "Bayesian MDS"),
  pch = c(19, 19), col = c("cyan", "magenta"))
par(opar)

## End(Not run)
```

`bmds_get_x_mode` *Posterior mode latent configuration in Bayesian multidimensional scaling (BMDS).*

Description

`bmds_get_x_mode` returns the latent configuration that produced the largest posterior value during the MCMC.

Usage

```
bmds_get_x_mode(D, res, p.i, min_p, max_p, start, end)
```

Arguments

`D`	Observed dissimilarities (provided as a distance matrix).
`res`	Results of a BMDS analysis as obtained with the `bmds` function.
`p.i`	A length-one numeric vector providing the index of the solution to use.
`min_p`	A length-one numeric vector providing the minimum value of the latent space dimension to use.
`max_p`	A length-one numeric vector providing the maximum value of the latent space dimension to use.
`start`	A length-one numeric vector providing the iteration number to start from.
`end`	A length-one numeric vector providing the iteration number where to end.

Value

A real matrix containing the posterior mode latent configuration.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

See Also

`bmds` for (one-way) Bayesian (metric) multidimensional scaling.

Examples

```
## Not run:
# Airline Distances Between Cities
airline <- read.csv(file = system.file("extdata", "airline.csv",
  package = "dmbc"))
airline.nm <- airline[, 1]
airline <- airline[, 2:31]
colnames(airline) <- airline.nm
airline <- as.dist(airline)
```

```
min_p <- 1
max_p <- 4
burnin <- 200
nsim <- 1000
totiter <- burnin + nsim

airline.mds <- cmdscale(airline, max_p)
airline.bmds <- bmds(airline, min_p, max_p, burnin, nsim)

opar <- par(mfrow = c(1, 2))
plot(min_p:max_p, airline.bmds$mdsIC$mdsic, type = "b",
  main = "MDS Information Criterion", xlab = "p", ylab = "MDSIC")
MDSICmin <- which.min(airline.bmds$mdsIC$mdsic)
points((min_p:max_p)[MDSICmin], airline.bmds$mdsIC$mdsic[MDSICmin],
  col = "red", pch = 10, cex = 1.75, lwd = 1.5)

airline.bmds.x.mode <- bmds_get_x_mode(airline, airline.bmds, MDSICmin,
  min_p, max_p, start = (burnin + 1), end = totiter)
airline.bmds.d <- dist(airline.bmds.x.mode)
airline.mds.d <- dist(airline.mds[, 1:((min_p:max_p)[MDSICmin])])
plot(airline, airline.bmds.d, type = "n", xlab = "observed",
  ylab = "estimated", main = "Airline Distances \n Between Cities",
  xlim = c(0, max(airline, airline.bmds.d)),
  ylim = c(0, max(airline, airline.bmds.d)))
abline(0, 1, lty = 2, col = "gray")
points(airline, airline.mds.d, pch = 19, col = "cyan", cex = .5)
points(airline, airline.bmds.d, pch = 19, col = "magenta", cex = .5)
legend(x = "bottomright", legend = c("Classical MDS", "Bayesian MDS"),
  pch = c(19, 19), col = c("cyan", "magenta"))
par(opar)

## End(Not run)
```

`check_list_na` *Auxiliary function to recursively check NAs in a list.*

Description

`check_list_na()` compares two lists and fills in the missing elements in the first with those included in the second. The comparison is recursive in the sense that the process is repeated for all lists included in those given.

Usage

```
check_list_na(orig, des)
```

Arguments

`orig`	A list whose content must be checked.
`des`	A list to use as a reference with which compare the first one.

Value

A list with all elements added.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

Examples

```
G <- 5
prior <- list(eta = list(a = rep(1, G), b = rep(2, G)))
check_list_na(prior, dmbc_prior())
```

`clusters,dmbc_config-method` *Extract the final cluster memberships from a* `dmbc_config` *class instance.*

Description

Extract the final cluster memberships from a `dmbc_config` class instance.

Usage

```
## S4 method for signature 'dmbc_config'
clusters(object, newdata = NULL, ...)
```

Arguments

`object`	An object of class `dmbc_config`.
`newdata`	An object of no explicit specification (currently ignored).
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

`comp_ssr` *Sum of squared residuals (SSR) from the observed distances and the given latent configuration.*

Description

`comp_ssr` computes the sum of squared residuals (SSR) from the observed distances (`diss`) and the given latent coordinates (`x`).

Usage

```
comp_ssr(x, diss)
```

Arguments

`x`	Real matrix containing the latent configuration.
`diss`	Observed dissimilarities (provided as a distance matrix).

Value

A length-one numeric vector providing the SSR for its arguments.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

See Also

`bmds` for (one-way) Bayesian (metric) multidimensional scaling.

Examples

```
n <- 10000
nr <- 200
nc <- floor(n/nr)
x <- matrix(rnorm(1:n), nrow = nr, ncol = nc)
obsdiss <- dist(x)
ssr <- numeric(ncol(x))
for (i in 1:ncol(x)) {
  ssr[i] <- comp_ssr(x[, 1:i], obsdiss)
}
plot(ssr, xlab = "number of dimensions", ylab = "SSR", type = "b")
```

`dmbc` *Estimation of a DMBC model.*

Description

`dmbc()`, the main function of the package, estimates a DMBC model for a given set of S dissimilarity matrices.

Usage

```
dmbc(
  data,
  p = 2,
  G = 3,
  control = dmbc_control(),
  prior = NULL,
  cl = NULL,
  post_all = FALSE
)
```

Arguments

`data`	An object of class `dmbc_data` containing the data to analyze.
`p`	A length-one numeric vector indicating the number of dimensions of the latent space.
`G`	A length-one numeric vector indicating the number of cluster to partition the S subjects.
`control`	A list of control parameters that affect the sampling but do not affect the posterior distribution. See `dmbc_control()` for more details.
`prior`	A list containing the prior hyperparameters. See `dmbc_prior()` for more details.
`cl`	An optional **parallel** or **snow** cluster for use if `parallel = "snow"`. If not supplied, a cluster on the local machine is created for the duration of the `dmbc()` call.
`post_all`	A length-one logical vector, which if TRUE applies a further post-processing to the simulated chains (in case these are more than one).

Value

A `dmbc_fit_list` object.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`bmds` for Bayesian (metric) multidimensional scaling.

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 20000
nsim <- 10000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

summary(sim.dmbc, include.burnin = FALSE)

library(bayesplot)
library(ggplot2)
color_scheme_set("teal")
plot(sim.dmbc, what = "trace", regex_pars = "eta")

z <- dmbc_get_configuration(sim.dmbc, chain = 1, est = "mean",
  labels = 1:16)
summary(z)
color_scheme_set("mix-pink-blue")
graph <- plot(z, size = 2, size_lbl = 3)
graph + panel_bg(fill = "gray90", color = NA)

## End(Not run)
```

`dmbc_check_groups` *Auxiliary function for checking the grouping results of a fitted DMBC model.*

Description

`dmbc_check_groups()` is an auxiliary function for checking whether the cluster membership estimates provided by the individual chains of the fitted model provided agree or not.

Usage

```
dmbc_check_groups(res, est = "mean")
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`est`	A length-one character vector indicating the estimate type to use.

Value

A length-one logical vector, which is equal to TRUE if all simulated chains provide the same cluster membership estimates, and FALSE otherwise.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_get_configuration()` for a description of the configuration extractor function.

`dmbc_fit_list` for a description of a fitted DMBC model.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
```

```
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

dmbc_check_groups(sim.dmbc)

## End(Not run)
```

`dmbc_config-class` *An S4 class to represent the latent configuration estimate for a DMBC model.*

Description

An S4 class to represent the the latent configuration estimate for a DMBC model.

Slots

- `Z.est` An array containing the estimate of the latent configuration for a DMBC model.
- `Z.sd` An array containing the standard deviation of the latent configuration for a DMBC model.
- `cluster` A numeric vector providing the estimated group membership for the *S* subjects in the data.
- `est` A length-one character vector providing the estimate type returned in `Z.est`. Possible values are `mean` (posterior mean), `median` (posterior median), `ml` (maximum likelihood) and `map` (maximum-a-posteriori).
- `n` A length-one numeric vector providing the number of objects.
- `p` A length-one numeric vector providing the number of latent dimensions.
- `S` A length-one numeric vector providing the number of subjects.
- `G` A length-one numeric vector providing the number of clusters.
- `family` An object of class `list`; named list with elements representing the parameter estimates corresponding to different values of *p* and *G*.
- `chain` A length-one numeric vector representing the ID of the MCMC chain used to compute the estimates.
- `labels` A character vector for the (optional) strings to use in the plots for labeling the objects.

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

Examples

```
showClass("dmbc_config")
```

`dmbc_control` *Auxiliary Function for Controlling DMBC Model Fitting*

Description

`dmbc_control()` is an auxiliary function as user interface for `dmbc()` fitting. Typically only used when calling the `dmbc()` function. It is used to set parameters that affect the sampling but do not affect the posterior distribution.

`control_dmbc()` is an alias for `dmbc_control()`.

`check_control()` is an auxiliary function that verifies the correctness of the controls provided before a DMBC is fitted with `dmbc()`.

Usage

```
dmbc_control(
  nsim = 5000,
  burnin = 10000,
  thin = 1,
  nchains = 1,
  threads = 1,
  seed = NULL,
  parallel = "no",
  z.prop = 1.5,
  alpha.prop = 0.75,
  random.start = TRUE,
  partition = NULL,
  method = "manhattan",
  procrustes = TRUE,
  relabel = TRUE,
  store.burnin = TRUE,
  verbose = FALSE
)

control_dmbc(
  nsim = 5000,
  burnin = 10000,
  thin = 1,
  nchains = 1,
  threads = 1,
  seed = NULL,
  parallel = "no",
  z.prop = 1.5,
```

```
  alpha.prop = 0.75,
  random.start = TRUE,
  partition = NULL,
  method = "manhattan",
  procrustes = TRUE,
  relabel = TRUE,
  store.burnin = TRUE,
  verbose = FALSE
)

check_control(control)
```

Arguments

`nsim`	A length-one numeric vector for the number of draws to be taken from the posterior distribution.
`burnin`	A length-one numeric vector for the number of initial MCMC iterations (usually to be discarded).
`thin`	A length-one numeric vector for the number of iterations between consecutive draws.
`nchains`	A length-one numeric vector for the number of parallel chains to run.
`threads`	A length-one numeric vector for the number of chains to run. If greater than 1, package **parallel** is used to take advantage of any multiprocessing or distributed computing capabilities that may be available.
`seed`	An integer scalar. If supplied, provides the random number seed.
`parallel`	A length-one character vector indicating the type of parallel operation to be used (if any). Possible values are `multicore` (which works only on Unix/mcOS), `snow` and `no` (i.e. serial instead of parallel computing).
`z.prop`	A length-one numeric vector providing the standard deviation of the proposal distribution for the jump in the individual latent space position.
`alpha.prop`	A length-one numeric vector providing the standard deviation of the proposal distribution for the jump in the individual random effect value.
`random.start`	A length-one logical vector. If `TRUE` the starting values are drawn randomly, otherwise a user-defined starting partition must be provided through the `partition` argument.
`partition`	A length-one numeric vector providing the user-defined starting partition.
`method`	A length-one character vector that specifies the distance measure to use in determining the initial partition. Allowed values are those from the `dist()` function.
`procrustes`	A length-one logical vector. If `TRUE` the simulated MCMC chains are post-processed through a Procrustes transformation.
`relabel`	A length-one logical vector. If `TRUE` the simulated MCMC chains are relabelled to address the label-switching problem.
`store.burnin`	A logical scalar. If `TRUE`, the samples from the burnin are also stored and returned.

`verbose`	A logical scalar. If `TRUE`, causes information to be printed out about the progress of the fitting.
`control`	A list of control options.

Value

A named list with the control options as components.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

See Also

`dmbc()`

Examples

```
## Not run:
data(simdiss, package = "dmbc")
# Shorter run than default.
sim.fit <- dmbc(simdiss,
  control = dmbc_control(burnin = 1000, nsim = 2000, thin = 5, verbose = TRUE))

## End(Not run)
```

`dmbc_data-class` *An S4 class to represent the data to use in a DMBC model.*

Description

An S4 class to represent the data to use in a DMBC model.

Slots

- `diss` A list whose elements are the dissimilarity matrices corresponding to the judgments expressed by the S subjects/raters. These matrices must be defined as a `dist` object.
- `n` A length-one character vector representing the number of objects compared by each subject.
- `S` A length-one numeric vector representing the number of subjects.
- `family` A length-one character vector representing the type of data to analyze. Currently, it accepts only the 'binomial' value, but future developments will include the possibility to analyze continuous, multinomial and count data.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

Examples

```
showClass("dmbc_data")
```

`dmbc_fit` *Fitter function for DMBC models.*

Description

`dmbc_fit()` is the main function that estimates a DMBC model.

Usage

```
dmbc_fit(D, p, G, family, control, prior, start)
```

Arguments

`D`	A list whose elements are the dissimilarity matrices corresponding to the judgments expressed by the S subjects/raters. These matrices must be defined as a `dist` object.
`p`	A length-one numeric vector indicating the number of dimensions of the latent space.
`G`	A length-one numeric vector indicating the number of cluster to partition the S subjects.
`family`	A length-one character vector representing the type of data to analyze. Currently, it accepts only the 'binomial' value, but future developments will include the possibility to analyze continuous, multinomial and count data.
`control`	A list of control parameters that affect the sampling but do not affect the posterior distribution See `dmbc_control()` for more details.
`prior`	A list containing the prior hyperparameters. See `dmbc_prior()` for more details.
`start`	A named list of starting values for the MCMC algorithm (see `dmbc_init`).

Value

A `dmbc_fit_list` object.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 20000
nsim <- 10000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

summary(sim.dmbc, include.burnin = FALSE)

library(bayesplot)
library(ggplot2)
color_scheme_set("teal")
plot(sim.dmbc, what = "trace", regex_pars = "eta")

z <- dmbc_get_configuration(sim.dmbc, chain = 1, est = "mean",
  labels = 1:16)
summary(z)
color_scheme_set("mix-pink-blue")
graph <- plot(z, size = 2, size_lbl = 3)
graph + panel_bg(fill = "gray90", color = NA)

## End(Not run)
```

`dmbc_fit-class` *An S4 class to represent the results of fitting DMBC model.*

Description

An S4 class to represent the results of fitting DMBC model using a single Markov Chain Monte Carlo chain.

Slots

`z.chain` An object of class `array`; posterior draws from the MCMC algorithm for the (untransformed) latent configuration Z.

`z.chain.p` An object of class `array`; posterior draws from the MCMC algorithm for the (Procrustes-transformed) latent configuration Z.

`alpha.chain` An object of class `matrix`; posterior draws from the MCMC algorithm for the α parameters.

`eta.chain` An object of class `matrix`; posterior draws from the MCMC algorithm for the η parameters.

`sigma2.chain` An object of class `matrix`; posterior draws from the MCMC algorithm for the σ^2 parameters.

`lambda.chain` An object of class `matrix`; posterior draws from the MCMC algorithm for the λ parameters.

`prob.chain` An object of class `array`; posterior draws from the MCMC algorithm for the cluster membership probabilities.

`x.ind.chain` An object of class `array`; posterior draws from the MCMC algorithm for the cluster membership indicators.

`x.chain` An object of class `matrix`; posterior draws from the MCMC algorithm for the cluster membership labels.

`accept` An object of class `matrix`; final acceptance rates for the MCMC algorithm.

`diss` An object of class `list`; list of observed dissimilarity matrices.

`dens` An object of class `list`; list of log-likelihood, log-prior and log-posterior values at each iteration of the MCMC simulation.

`control` An object of class `list`; list of the control parameters (number of burnin and sample iterations, number of MCMC chains, etc.). See `dmbc_control()` for more information.

`prior` An object of class `list`; list of the prior hyperparameters. See `dmbc_prior()` for more information.

`dim` An object of class `list`; list of dimensions for the estimated model, i.e. number of objects (*n*), number of latent dimensions (*p*), number of clusters (*G*), and number of subjects (*S*).

`model` An object of class `dmbc_model`.

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

Examples

```
showClass("dmbc_fit")
```

`dmbc_fit_list-class` *An S4 class to represent the results of fitting DMBC model.*

Description

An S4 class to represent the results of fitting DMBC model using multiple Markov Chain Monte Carlo chains.

Slots

`results` An object of class `list`; list of `dmbc_fit` objects corresponding to the parallel MCMC chains simulated during the estimation.

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_fit` for more details on the components of each element of the list.

Examples

```
showClass("dmbc_fit_list")
```

`dmbc_fit_list_to_list` *Conversion of an* `dmbc_fit_list` *object to a* `list`.

Description

`dmbc_fit_list_to_list` converts an object of class `dmbc_fit_list` to a list of arrays including all the parameter. chains. It is intended for internal use mainly.

Usage

```
dmbc_fit_list_to_list(res, include.burnin = FALSE, verbose = TRUE)
```

Arguments

`res`	An object of type `dmbc_fit_list`.
`include.burnin`	A logical scalar. If `TRUE` the burnin iterations (if available) are not removed.
`verbose`	A logical scalar. If `TRUE` prints additional warnings during the conversion.

Value

An object of type `mcmc.list`.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

See Also

`dmbc()` for for fitting a DMBC model; `dmbc_fit_list-class`.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], nchains = 2, verbose = TRUE)
sim.dmbc <- dmbc(simdiss, p, G, control)
sim.list <- dmbc_fit_list_to_list(sim.dmbc, TRUE)
```

```
library(bayesplot)
mcmc_trace(sim.list, regex_pars = "lambda")

## End(Not run)
```

`dmbc_fit_list_to_mcmc.list` *Conversion of an* `dmbc_fit_list` *object to an object of class* `mcmc.list`*.*

Description

`dmbc_fit_list_to_mcmc.list` converts an object of class `dmbc_fit_list` to an object with class `mcmc.list`.

Usage

```
dmbc_fit_list_to_mcmc.list(res, include.burnin = FALSE, verbose = TRUE)
```

Arguments

`res`	An object of type `dmbc_fit_list`.
`include.burnin`	A logical scalar. If `TRUE` the burnin iterations (if available) are not removed.
`verbose`	A logical scalar. If `TRUE` prints additional warnings during the conversion.

Value

An object of type `mcmc.list`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

See Also

`dmbc()` for for fitting a DMBC model; `dmbc_fit_list-class`; `mcmc.list`.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301
```

```
set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], nchains = 2, verbose = TRUE)
sim.dmbc <- dmbc(simdiss, p, G, control)
sim.mcmc <- dmbc_fit_list_to_mcmc.list(sim.dmbc, TRUE)
plot(sim.mcmc)

## End(Not run)
```

`dmbc_fit_to_mcmc` *Conversion of an* `dmbc_fit` *object to an object of class* `mcmc`.

Description

`dmbc_fit_to_mcmc` converts an object of class `dmbc_fit` to an object with class `mcmc`.

Usage

```
dmbc_fit_to_mcmc(res, include.burnin = FALSE, verbose = TRUE)
```

Arguments

`res`	An object of type `dmbc_fit`.
`include.burnin`	A logical scalar. If `TRUE` the burnin iterations (if available) are not removed.
`verbose`	A logical scalar. If `TRUE` prints additional warnings during the conversion.

Value

An object of type `mcmc`.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

See Also

`dmbc()` for for fitting a DMBC model; `dmbc_fit-class`; `mcmc`.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
```

```
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], verbose = TRUE)
sim.dmbc <- dmbc(simdiss, p, G, control)
sim.mcmc <- dmbc_fit_to_mcmc(sim.dmbc@results[[1]], TRUE)
plot(sim.mcmc)

## End(Not run)
```

`dmbc_get_configuration` *Extractor function for a fitted DMBC model.*

Description

`dmbc_get_configuration()` is an extractor function for extracting the latent configuration estimates of a fitted DMBC model.

Usage

```
dmbc_get_configuration(res, chain = 1, est = "mean", labels = character(0))
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`chain`	A length-one numeric vector indicating the MCMC chain number to use.
`est`	A length-one character vector indicating the estimate type to use.
`labels`	An optional character vector with the object labels.

Value

A `dmbc_config` object.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

z <- dmbc_get_configuration(sim.dmbc, chain = 1, est = "mean")
summary(z)

library(bayesplot)
library(ggplot2)
color_scheme_set("mix-pink-blue")
graph <- plot(z, size = 2, size_lbl = 3)
graph + panel_bg(fill = "gray90", color = NA)

## End(Not run)
```

`dmbc_get_map` *Extractor function for a fitted DMBC model.*

Description

`dmbc_get_map()` is an extractor function for extracting the maximum-a-posterior estimates of the parameters for a fitted DMBC model.

Usage

```
dmbc_get_map(res, chain = 1)
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`chain`	A length-one numeric vector indicating the MCMC chain number to use.

Value

A named `list` with the following elements:

`z`**:** array of latent coordinates posterior mean estimates

`alpha`**:** numeric vector of alpha posterior mean estimates

`eta`**:** numeric vector of eta posterior mean estimates

`sigma2`**:** numeric vector of sigma2 posterior mean estimates

`lambda`**:** numeric vector of lambda posterior mean estimates

`prob`**:** numeric matrix of probability posterior mean estimates

`cluster`**:** numeric vector of cluster membership posterior mean estimates

`logpost`**:** length-one numeric vector of the maximum log-posterior value

`chain`**:** length-one numeric vector of the MCMC chain number used

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)
```

```
control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

dmbc_get_map(sim.dmbc, chain = 1)

## End(Not run)
```

`dmbc_get_ml` *Extractor function for a fitted DMBC model.*

Description

`dmbc_get_ml()` is an extractor function for extracting the maximum likelihood estimates of the parameters for a fitted DMBC model.

Usage

```
dmbc_get_ml(res, chain = 1)
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`chain`	A length-one numeric vector indicating the MCMC chain number to use.

Value

A named `list` with the following elements:

`z`: array of latent coordinates posterior mean estimates

`alpha`: numeric vector of alpha posterior mean estimates

`eta`: numeric vector of eta posterior mean estimates

`sigma2`: numeric vector of sigma2 posterior mean estimates

`lambda`: numeric vector of lambda posterior mean estimates

`prob`: numeric matrix of probability posterior mean estimates

`cluster`: numeric vector of cluster membership posterior mean estimates

`loglik`: length-one numeric vector of the maximum log-likelihood value

`chain`: length-one numeric vector of the MCMC chain number used

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

dmbc_get_ml(sim.dmbc, chain = 1)

## End(Not run)
```

`dmbc_get_postmean` *Extractor function for a fitted DMBC model.*

Description

`dmbc_get_postmean()` is an extractor function for extracting the posterior mean estimates of the parameters for a fitted DMBC model.

Usage

```
dmbc_get_postmean(res, chain = 1)
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`chain`	A length-one numeric vector indicating the MCMC chain number to use.

Value

A named `list` with the following elements:

`z`**:** array of latent coordinates posterior mean estimates

`alpha`**:** numeric vector of alpha posterior mean estimates

`eta`**:** numeric vector of eta posterior mean estimates

`sigma2`**:** numeric vector of sigma2 posterior mean estimates

`lambda`**:** numeric vector of lambda posterior mean estimates

`prob`**:** numeric matrix of probability posterior mean estimates

`cluster`**:** numeric vector of cluster membership posterior mean estimates

`chain`**:** length-one numeric vector of the MCMC chain number used

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
```

```
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

dmbc_get_postmean(sim.dmbc, chain = 1)

## End(Not run)
```

`dmbc_get_postmedian` *Extractor function for a fitted DMBC model.*

Description

`dmbc_get_postmedian()` is an extractor function for extracting the posterior median estimates of the parameters for a fitted DMBC model.

Usage

```
dmbc_get_postmedian(res, chain = 1)
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`chain`	A length-one numeric vector indicating the MCMC chain number to use.

Value

A named `list` with the following elements:

`z`: array of latent coordinates posterior median estimates

`alpha`: numeric vector of alpha posterior median estimates

`eta`: numeric vector of eta posterior median estimates

`sigma2`: numeric vector of sigma2 posterior median estimates

`lambda`: numeric vector of lambda posterior median estimates

`prob`: numeric matrix of probability posterior median estimates

`cluster`: numeric vector of cluster membership posterior median estimates

`chain`: length-one numeric vector of the MCMC chain number used

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_data` for a description of the data format.

`dmbc_fit_list` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 3
p <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 2, thin = 10, store.burnin = TRUE, threads = 2,
  parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

dmbc_get_postmedian(sim.dmbc, chain = 1)

## End(Not run)
```

`dmbc_IC` *Model selection of DMBC models.*

Description

`dmbc_IC()` is the main function for simultaneously selecting the optimal latent space dimension (p) and number of clusters (G) for a DMBC analysis.

Usage

```
dmbc_IC(
  data,
  pmax = 3,
  Gmax = 5,
  control = dmbc_control(),
  prior = NULL,
  est = "mean"
)
```

Arguments

`data`	An object of class `dmbc_data` containing the data to analyze.
`pmax`	A length-one numeric vector indicating the maximum number of dimensions of the latent space to consider.
`Gmax`	A length-one numeric vector indicating the maximum number of cluster to consider.
`control`	A list of control parameters that affect the sampling but do not affect the posterior distribution See `dmbc_control()` for more details.
`prior`	A list containing the prior hyperparameters. See `dmbc_prior()` for more details.
`est`	A length-one character vector indicating the estimate type to use. Possible values are `mean`, `median`, `ml` and `map`.

Value

A `dmbc_ic` object.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc()` for fitting a DMBC model.

`dmbc_ic` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = ”dmbc”)

pmax <- 2
Gmax <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 1809

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[[”z”]],
  alpha.prop = prm.prop[[”alpha”]], random.start = TRUE, verbose = TRUE,
  thin = 10, store.burnin = TRUE)
```

```
sim.ic <- dmbc_IC(data = simdiss, pmax = pmax, Gmax = Gmax, control = control,
  est = "mean")

pmax <- pmax + 1
Gmax <- Gmax + 2
new.ic <- update(sim.ic, pmax = pmax, Gmax = Gmax)
new.ic

# plot the results
library(bayesplot)
library(ggplot2)
color_scheme_set("mix-yellow-blue")
p <- plot(new.ic, size = c(4, 1.5))
p + panel_bg(fill = "gray90", color = NA)

## End(Not run)
```

`dmbc_ic-class` *An S4 class to represent the comparison of a set of DMBC models.*

Description

An S4 class to represent the comparison of a set of DMBC models through the dissimilarity model-based clustering information criterion (DCIC).

Slots

- `logprior` An object of class `matrix` providing the log-prior values corresponding to different values of p and G.
- `logmlik` An object of class `matrix` providing the marginal log-likelihood values corresponding to different values of p and G.
- `logcorrfact` An object of class `matrix` providing the logarithm of the correction factors corresponding to different values of p and G.
- `DCIC` An object of class `matrix` providing the values of the DCIC index corresponding to different values of p and G.
- `post.est` An object of class `list`; named list with elements representing the parameter estimates corresponding to different values of p and G.
- `est` A length-one character vector representing the estimate type used in computing the DCIC index. Possible values are `mean`, `median`, `ml` and `map`. See `dmbc_ic()` for more details about these values.
- `res_last_p` An object of class `list`; list of `dmbc_fit_list` objects with the results of fitting the DMBC models corresponding to the last value of p. This is needed in case of an update of the DCIC calculations using additional p and/or G values.

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

Examples

```
showClass("dmbc_ic")
```

`dmbc_init` *Function to compute the starting values before fitting a DMBC models.*

Description

`dmbc_init()` is the main function that estimates a DMBC model.

Usage

```
dmbc_init(D, p, G, family, random.start, method, partition)
```

Arguments

`D`	A list whose elements are the dissimilarity matrices corresponding to the judgments expressed by the S subjects/raters. These matrices must be defined as a `dist` object.
`p`	A length-one numeric vector indicating the number of dimensions of the latent space.
`G`	A length-one numeric vector indicating the number of cluster to partition the S subjects.
`family`	A length-one character vector representing the type of data to analyze. Currently, it accepts only the 'binomial' value, but future developments will include the possibility to analyze continuous, multinomial and count data.
`random.start`	A length-one logical vector. If `TRUE` the starting values are drawn randomly, otherwise.
`method`	A length-one character vector specifying the distance measure to use in determining the initial partition. Allowed values are those from the `dist()` function.
`partition`	A length-one numeric vector providing the user-defined starting partition.

Value

A named `list` with the following items:

- `z`: array of latent coordinates starting values
- `x`: numeric vector of initial cluster memberships
- `ng`: numeric vector of initial cluster sizes
- `alpha`: numeric vector of alpha starting values
- `eta`: numeric vector of eta starting values
- `sigma2`: numeric vector of sigma2 starting values
- `lambda`: numeric vector of lambda starting values

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc()` for fitting a DMBC model.

Examples

```
data(simdiss, package = "dmbc")
dmbc_init(simdiss@diss, p = 2, G = 3, family = "binomial", random.start = TRUE)
```

`dmbc_logLik` *Log-likelihood for DMBC models.*

Description

`dmbc_logLik()` computes the log-likelihood value for a DMBC model.

Usage

```
dmbc_logLik(D, Z, alpha, lambda, x)
```

Arguments

`D`	A list whose elements are the dissimilarity matrices corresponding to the judgments expressed by the S subjects/raters. These matrices must be defined as a `dist` object.
`Z`	A numeric matrix containing the latent configuration.
`alpha`	A numeric vector containing the alpha values.
`lambda`	A numeric vector containing the alpha lambda.
`x`	A numeric vector containing the cluster indicator values.

Value

A length-one numeric vector of the log-likelihood value.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc()`.

`dmbc_logLik_rbmds` *Log-likelihood for DMBC models.*

Description

`dmbc_logLik_rbmds()` computes the log-likelihood value for a DMBC model.

Usage

```
dmbc_logLik_rbmds(D, Z, alpha)
```

Arguments

`D`	A list whose elements are the dissimilarity matrices corresponding to the judgments expressed by the S subjects/raters. These matrices must be defined as a `dist` object.
`Z`	A numeric matrix containing the latent configuration.
`alpha`	A numeric vector containing the alpha values.

Value

A length-one numeric vector of the log-likelihood value.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc()`.

`dmbc_match_groups` *Auxiliary function for realigning the grouping of a fitted DMBC model.*

Description

`dmbc_match_groups()` is an auxiliary function for realigning the cluster membership estimates provided by the individual chains of the fitted model if they do not agree.

Usage

```
dmbc_match_groups(res, est = "mean", ref = 1)
```

Arguments

`res`	An object of class `dmbc_fit_list`.
`est`	A length-one character vector indicating the estimate type to use.
`ref`	A length-one numeric vector indicating the chain number to use as the reference.

Value

An object of class `dmbc_fit_list`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc_check_groups()` for checking the consistency of the cluster memberships across chains for a fitted DMBC model.

`dmbc_get_configuration()` for a description of the configuration extractor function.

`dmbc_fit_list` for a description of a fitted DMBC model.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

G <- 5
p <- 3
prm.prop <- list(z = 4, alpha = 2)
burnin <- 2000
nsim <- 1000
seed <- 2301

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  nchains = 6, store.burnin = TRUE, threads = 2, parallel = "snow")
sim.dmbc <- dmbc(simdiss, p, G, control)

sim.dmbc_new <- dmbc_match_groups(sim.dmbc)

## End(Not run)
```

`dmbc_model-class` *An S4 class to represent a DMBC model.*

Description

An S4 class to represent a DMBC model.

Slots

- `p` A length-one character vector representing the number of dimensions of the latent space to use in the MDS analysis.
- `G` A length-one numeric vector representing the number of clusters to partition the subjects into.
- `family` A length-one character vector representing the type of data to analyze. Currently, it accepts only the 'binomial' value, but future developments will include the possibility to analyze continuous, multinomial and count data.

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

Examples

```
showClass("dmbc_model")
```

`dmbc_prior` *Auxiliary Function for Setting DMBC Model Priors*

Description

`dmbc_prior()` is an auxiliary function as user interface for `dmbc()` fitting. Typically only used when calling the `dmbc()` function. It is used to set prior hyperparameters.

`prior_dmbc()` is an alias for `dmbc_prior()`.

`check_prior()` is an auxiliary function that verifies the correctness of the prior hyperparameters provided before a DMBC is fitted with `dmbc()`.

`update_prior()` is an auxiliary function to modify a set of prior choices using a new value of p and G. It is intended for internal use mainly in the `dmbc_ic()` function.

Usage

```
dmbc_prior(
  eta = list(a = rep(1.5, .dmbcEnv$current_G), b = rep(0.5, .dmbcEnv$current_G)),
   sigma2 = list(a = 0.1, b = 0.1),
   lambda = rep(1, .dmbcEnv$current_G)
)

prior_dmbc(
  eta = list(a = rep(1.5, .dmbcEnv$current_G), b = rep(0.5, .dmbcEnv$current_G)),
   sigma2 = list(a = 0.1, b = 0.1),
   lambda = rep(1, .dmbcEnv$current_G)
)

check_prior(prior)

update_prior(prior, p, G)
```

Arguments

`eta`	A named list containing the hyperparameters for the prior distribution of the $\eta_1, \ldots, \eta_G$ parameters. It should contain two numeric vectors, namely `a` and `b`.

`sigma2`	A named list containing the hyperparameters for the prior distributions of the $\sigma_1^2, \ldots, \sigma_G^2$ parameters. It should contain two numeric scalars, namely `a` and `b`.
`lambda`	A list containing the hyperparameters for the prior distribution of the $\lambda_1, \ldots, \lambda_G$ parameters. It should contain a single numeric vector.
`prior`	A named list of prior hyperparameters.
`p`	A length-one numeric vector indicating the number of dimensions of the latent space.
`G`	A length-one numeric vector indicating the number of cluster to partition the S subjects.

Value

A list with the prior hyperparameters as components.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

See Also

`dmbc()`

Examples

```
## Not run:
data(simdiss, package = "dmbc")
# Shorter run than default.
sim.fit <- dmbc(simdiss,
  control = dmbc_control(burnin = 1000, nsim = 2000, thin = 1, verbose = TRUE),
  prior = dmbc_prior(sigma2 = list(a = 1, b = 4)))

## End(Not run)
```

`initialize,dmbc_config-method`

Create an instance of the `dmbc_config` *class using new/initialize.*

Description

Create an instance of the `dmbc_config` class using new/initialize.

Usage

```
## S4 method for signature 'dmbc_config'
initialize(
  .Object,
  Z.est = array(),
  Z.sd = array(),
  cluster = numeric(),
  est = character(),
  n = numeric(),
  S = numeric(),
  p = numeric(),
  G = numeric(),
  family = character(),
  chain = numeric(),
  labels = character()
)
```

Arguments

`.Object`	Prototype object from the class `dmbc_config`.
`Z.est`	An array containing the estimate of the latent configuration for a DMBC model.
`Z.sd`	An array containing the standard deviation of the latent configuration for a DMBC model.
`cluster`	A numeric vector providing the estimated group membership for the S subjects in the data.
`est`	A length-one character vector providing the estimate type returned in `Z.est`. Possible values are `mean` (posterior mean), `median` (posterior median), `ml` (maximum likelihood) and `map` (maximum-a-posteriori).
`n`	A length-one numeric vector providing the number of objects.
`S`	A length-one numeric vector providing the number of subjects.
`p`	A length-one numeric vector providing the number of latent dimensions.
`G`	A length-one numeric vector providing the number of clusters.
`family`	An object of class `list`; named list with elements representing the parameter estimates corresponding to different values of p and G.
`chain`	A length-one numeric vector representing the ID of the MCMC chain used to compute the estimates.
`labels`	A character vector for the (optional) strings to use in the plots for labeling the objects.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

`initialize,dmbc_data-method`
Create an instance of the `dmbc_data` *class using new/initialize.*

Description

Create an instance of the `dmbc_data` class using new/initialize.

Usage

```
## S4 method for signature 'dmbc_data'
initialize(
  .Object,
  diss = list(),
  n = numeric(),
  S = numeric(),
  family = character()
)
```

Arguments

`.Object`	Prototype object from the class `dmbc_data`.
`diss`	A list whose elements are the dissimilarity matrices corresponding to the judgments expressed by the *S* subjects/raters. These matrices must be defined as a `dist` object.
`n`	A length-one character vector representing the number of objects compared by each subject.
`S`	A length-one numeric vector representing the number of subjects.
`family`	A length-one character vector representing the type of data to analyze. Currently, it accepts only the 'binomial' value, but future developments will include the possibility to analyze continuous, multinomial and count data.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

`initialize,dmbc_fit-method`
Create an instance of the `dmbc_fit` *class using new/initialize.*

Description

Create an instance of the `dmbc_fit` class using new/initialize.

Usage

```
## S4 method for signature 'dmbc_fit'
initialize(
  .Object,
  z.chain = array(),
  z.chain.p = array(),
  alpha.chain = matrix(),
  eta.chain = matrix(),
  sigma2.chain = matrix(),
  lambda.chain = matrix(),
  prob.chain = array(),
  x.ind.chain = array(),
  x.chain = matrix(),
  accept = matrix(),
  diss = list(),
  dens = list(),
  control = list(),
  prior = list(),
  dim = list(),
  model = NA
)
```

Arguments

`.Object`	Prototype object from the class `dmbc_fit`.
`z.chain`	An object of class `array`; posterior draws from the MCMC algorithm for the (untransformed) latent configuration Z.
`z.chain.p`	An object of class `array`; posterior draws from the MCMC algorithm for the (Procrustes-transformed) latent configuration Z.
`alpha.chain`	An object of class `matrix`; posterior draws from the MCMC algorithm for the α parameters.
`eta.chain`	An object of class `matrix`; posterior draws from the MCMC algorithm for the η parameters.
`sigma2.chain`	An object of class `matrix`; posterior draws from the MCMC algorithm for the σ^2 parameters.
`lambda.chain`	An object of class `matrix`; posterior draws from the MCMC algorithm for the λ parameters.
`prob.chain`	An object of class `array`; posterior draws from the MCMC algorithm for the cluster membership probabilities.
`x.ind.chain`	An object of class `array`; posterior draws from the MCMC algorithm for the cluster membership indicators.
`x.chain`	An object of class `matrix`; posterior draws from the MCMC algorithm for the cluster membership labels.
`accept`	An object of class `matrix`; final acceptance rates for the MCMC algorithm.
`diss`	An object of class `list`; list of observed dissimilarity matrices.

`dens`	An object of class `list`; list of log-likelihood, log-prior and log-posterior values at each iteration of the MCMC simulation.
`control`	An object of class `list`; list of the control parameters (number of burnin and sample iterations, number of MCMC chains, etc.). See `dmbc_control()` for more information.
`prior`	An object of class `list`; list of the prior hyperparameters. See `dmbc_prior()` for more information.
`dim`	An object of class `list`; list of dimensions for the estimated model, i.e. number of objects (n), number of latent dimensions (p), number of clusters (G), and number of subjects (S).
`model`	An object of class `dmbc_model`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`initialize,dmbc_fit_list-method`

Create an instance of the `dmbc_fit_list` *class using new/initialize.*

Description

Create an instance of the `dmbc_fit_list` class using new/initialize.

Usage

```
## S4 method for signature 'dmbc_fit_list'
initialize(.Object, results = list())
```

Arguments

`.Object`	Prototype object from the class `dmbc_fit_list`.
`results`	A list whose elements are the `dmbc_fit` objects for each simulated chain.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`initialize,dmbc_ic-method`

Create an instance of the `dmbc_ic` *class using new/initialize.*

Description

Create an instance of the `dmbc_ic` class using new/initialize.

Usage

```
## S4 method for signature 'dmbc_ic'
initialize(
  .Object,
  logprior = matrix(),
  logmlik = matrix(),
  logcorrfact = matrix(),
  DCIC = matrix(),
  post.est = list(),
  est = character(),
  res_last_p = list()
)
```

Arguments

`.Object`	Prototype object from the class `dmbc_ic`.
`logprior`	An object of class `matrix` providing the log-prior values corresponding to different values of p and G.
`logmlik`	An object of class `matrix` providing the marginal log-likelihood values corresponding to different values of p and G.
`logcorrfact`	An object of class `matrix` providing the logarithm of the correction factors corresponding to different values of p and G.
`DCIC`	An object of class `matrix` providing the values of the DCIC index corresponding to different values of p and G.
`post.est`	An object of class `list`; named list with elements representing the parameter estimates corresponding to different values of p and G.
`est`	A length-one character vector representing the estimate type used in computing the DCIC index. Possible values are `mean`, `median`, `ml` and `map`. See `dmbc_ic()` for more details about these values.
`res_last_p`	An object of class `list`; list of `dmbc_fit_list` objects with the results of fitting the DMBC models corresponding to the last value of p. This is needed in case of an update of the DCIC calculations using additional p and/or G values.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`initialize,dmbc_model-method` *Create an instance of the* `dmbc_model` *class using new/initialize.*

Description

Create an instance of the `dmbc_model` class using new/initialize.

Usage

```
## S4 method for signature 'dmbc_model'
initialize(.Object, p = numeric(), G = numeric(), family = character())
```

Arguments

`.Object`	Prototype object from the class `dmbc_model`.
`p`	A length-one character vector representing the number of dimensions of the latent space to use in the MDS analysis.
`G`	A length-one numeric vector representing the number of clusters to partition the subjects into.
`family`	A length-one character vector representing the type of data to analyze. Currently, it accepts only the ’binomial’ value, but future developments will include the possibility to analyze continuous, multinomial and count data.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`kinship` *List of binary dissimilarity matrices among 15 kinship terms.*

Description

Rosenberg and Kim (1975) designed an experiment to analyze the perceived similarities of 15 kinship terms.

Here, we consider the data relative to 85 females made available in Rosenberg (1982). Each subject was asked to group the kinship terms according to the perceived similarity. Thus, $S = 85$ binary dissimilarity matrices are available whose elements (0 or 1) indicate whether or not two kinship terms were grouped together by each individual.

Usage

```
data(kinship)
```

Format

A `dmbc_data` object whose `diss` element is a list of 85 binary dissimilarity matrices. Each matrix is defined as a `dist` object measuring whether each pair of the 15 kinship terms is judged as similar (1) or not (0).

The `dist` objects have rows and columns that are named as follows:

GrF	grandfather
GrM	grandmother
GrD	granddaughter
GrS	grandson
Bro	brother
Sis	sister
Fat	father
Mot	mother
Dau	daughter
Son	son
Nep	nephew
Nie	niece
Cou	cousin
Aun	aunt
Unc	uncle

References

Rosenberg, S. (1982). The method of sorting in multivariate research with applications selected from cognitive psychology and person perception. In N Hirschberg, LG Humphreys (eds.), Multivariate Applications in the Social Sciences, pp. 117–142. Erlbaum., Hillsdale, NJ.

Rosenberg, S., Kim, M. P. (1975). The method of sorting as a data-gathering procedure in multivariate research. Multivariate Behavioral Research, 10.

Examples

```
data(kinship)
library(bayesplot)
cols <- color_scheme_set("mix-red-blue")
plot(kinship, colors = unlist(cols)[c(1, 6)], font = 1, cex.font = 0.75)
```

`mdsic` *Information criterion for Bayesian multidimensional scaling (BMDS).*

Description

`mdsic` computes the information criterion for a set of Bayesian multidimensional scaling (BMDS) solutions using the approach in Oh & Raftery (2001).

Usage

```
mdsic(x_star, rmin_ssr, n, min_p = 1, max_p = 6)
```

Arguments

`x_star`	An array containing the latent configurations estimated using `bmds`.
`rmin_ssr`	A numeric vector providing the ratios of SSR for the latent dimensions requested.
`n`	A length-one numeric vector providing the number of objects.
`min_p`	A length-one numeric vector providing the minimum value of the latent space dimension to use.
`max_p`	A length-one numeric vector providing the maximum value of the latent space dimension to use.

Value

A list with the following elements:

`mdsic` A numeric vector with the values of MDSIC index.

`bic` A numeric vector with the values of the BIC index.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Oh, M.-S., Raftery, A. E. (2001), "Bayesian Multidimensional Scaling and Choice of Dimension", Journal of the American Statistical Association, 96, 1031-1044.

See Also

`bmds` for Bayesian (metric) multidimensional scaling and `comp_ssr` for the computation of SSR.

Examples

```
## Not run:
# Road distances (in km) between 21 cities in Europe
data(eurodist, package = "datasets")

min_p <- 1
max_p <- 10
burnin <- 200
nsim <- 1000
totiter <- burnin + nsim

eurodist.mds <- cmdscale(eurodist, max_p)
eurodist.bmds <- bmds(eurodist, min_p, max_p, burnin, nsim)

plot((min_p:max_p), eurodist.bmds$mdsIC$mdsic, type = "b",
  main = "MDS Information Criterion", xlab = "p", ylab = "MDSIC")
MDSICmin <- which.min(eurodist.bmds$mdsIC$mdsic)
points((min_p:max_p)[MDSICmin], eurodist.bmds$mdsIC$mdsic[MDSICmin],
  col = "red", pch = 10, cex = 1.75, lwd = 1.5)

## End(Not run)
```

`plot,dmbc_config,ANY-method`

Provide a graphical summary of a `dmbc_config` *class instance.*

Description

Provide a graphical summary of a `dmbc_config` class instance.

Usage

```
## S4 method for signature 'dmbc_config,ANY'
plot(
  x,
  size = NULL,
  size_lbl = NULL,
  nudge_x = 0,
  nudge_y = 0,
  label_objects = TRUE,
  ...
)
```

Arguments

`x`	An object of class `dmbc_config`.
`size`	A length-two numeric vector providing the optional sizes of points and lines in the plot.

`size_lbl`	A length-one numeric vector providing the size of labels.
`nudge_x`	A length-one numeric vector providing the optional horizontal adjustment to nudge labels by.
`nudge_y`	A length-one numeric vector providing the optional vertical adjustment to nudge labels by.
`label_objects`	A length-one logical vector. If `TRUE`, labels are added to the plot.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`plot,dmbc_data,ANY-method`

Provide a graphical summary of a `dmbc_data` *class instance.*

Description

Provide a graphical summary of a `dmbc_data` class instance.

Usage

```
## S4 method for signature 'dmbc_data,ANY'
plot(x, colors = c("white", "black"), font = NA, cex.font = NA, ...)
```

Arguments

`x`	An object of class `dmbc_data`.
`colors`	A character vector providing the colors to use in the plot.
`font`	A length-one numeric vector for the font to use for text. Can be a vector. `NA` values (the default) mean use `par("font")`.
`cex.font`	A length-one numeric vector for the character expansion factor. `NULL` and `NA` are equivalent to `1.0`. This is an absolute measure, not scaled by `par("cex")` or by setting ' `par("mfrow")` or `par("mfcol")`. Can be a vector.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

Examples

```
data(simdiss)
library(bayesplot)
cols <- color_scheme_set("brightblue")
plot(simdiss, colors = unlist(cols)[c(1, 6)], font = 1, cex.font = 0.75)
```

`plot,dmbc_fit,ANY-method`

Provide a graphical summary of a `dmbc_fit` *class instance.*

Description

Provide a graphical summary of a `dmbc_fit` class instance.

Usage

```
## S4 method for signature 'dmbc_fit,ANY'
plot(
  x,
  what = "trace",
  pars = character(),
  regex_pars = "lambda",
  include.burnin = FALSE,
  combo = NULL,
  ...
)
```

Arguments

`x`	An object of class `dmbc_fit`.
`what`	A length-one character vector providing the plot type to produce. Admissible values are those provided by the **bayesplot** package, that is: `acf`, `areas`, `dens`, `hex`, `hist`, `intervals`, `neff`, `pairs`, `parcoord`, `recover`, `rhat`, `scatter`, `trace`, `violin` or `combo`. In particular, `combo` allows to mix different plot types. For more details see the documentation of the **bayesplot** package.
`pars`	An optional character vector of parameter names. If neither `pars` nor `regex_pars` is specified, the default is to use all parameters.
`regex_pars`	An optional `regular expression` to use for parameter selection. Can be specified instead of `pars` or in addition to `pars`.
`include.burnin`	A length-one logical vector. If `TRUE` the burnin iterations (if available) are included in the summary.
`combo`	A character vector providing the plot types to combine (see `mcmc_combo`).
`...`	Further arguments to pass on.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

---

plot,dmbc_fit_list,ANY-method

Provide a graphical summary of a `dmbc_fit_list` *class instance.*

Description

Provide a graphical summary of a `dmbc_fit_list` class instance.

Usage

```
## S4 method for signature 'dmbc_fit_list,ANY'
plot(
  x,
  what = "trace",
  pars = character(),
  regex_pars = "lambda",
  include.burnin = FALSE,
  combo = NULL,
  ...
)
```

Arguments

`x`	An object of class `dmbc_fit_list`.
`what`	A length-one character vector providing the plot type to produce. Admissible values are those provided by the **bayesplot** package, that is: `acf`, `areas`, `dens`, `hex`, `hist`, `intervals`, `neff`, `pairs`, `parcoord`, `recover`, `rhat`, `scatter`, `trace`, `violin` or `combo`. In particular, `combo` allows to mix different plot types. For more details see the documentation of the **bayesplot** package.
`pars`	An optional character vector of parameter names. If neither `pars` nor `regex_pars` is specified, the default is to use all parameters.
`regex_pars`	An optional `regular expression` to use for parameter selection. Can be specified instead of `pars` or in addition to `pars`.
`include.burnin`	A length-one logical vector. If `TRUE` the burnin iterations (if available) are included in the summary.
`combo`	A character vector providing the plot types to combine (see `mcmc_combo`).
`...`	Further arguments to pass on.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`plot,dmbc_ic,ANY-method`

Provide a graphical summary of a `dmbc_ic` *class instance.*

Description

Provide a graphical summary of a `dmbc_ic` class instance.

Usage

```
## S4 method for signature 'dmbc_ic,ANY'
plot(x, size = NULL, ...)
```

Arguments

`x`	An object of class `dmbc_ic`.
`size`	A length-two numeric vector providing the optional sizes of points and lines in the plot.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`show,dmbc_config-method`

Show an instance of the `dmbc_config` *class.*

Description

Show an instance of the `dmbc_config` class.

Usage

```
## S4 method for signature 'dmbc_config'
show(object)
```

Arguments

`object`	An object of class `dmbc_config`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`show,dmbc_data-method` *Show an instance of the* `dmbc_data` *class.*

Description

Show an instance of the `dmbc_data` class.

Usage

```
## S4 method for signature 'dmbc_data'
show(object)
```

Arguments

`object`	An object of class `dmbc_data`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`show,dmbc_fit-method` *Show an instance of the* `dmbc_fit` *class.*

Description

Show an instance of the `dmbc_fit` class.

Usage

```
## S4 method for signature 'dmbc_fit'
show(object)
```

Arguments

`object`	An object of class `dmbc_fit`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`show,dmbc_fit_list-method` *Show an instance of the* `dmbc_fit_list` *class.*

Description

Show an instance of the `dmbc_fit_list` class.

Usage

```
## S4 method for signature 'dmbc_fit_list'
show(object)
```

Arguments

`object`	An object of class `dmbc_fit_list`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`show,dmbc_ic-method` *Show an instance of the* `dmbc_ic` *class.*

Description

Show an instance of the `dmbc_ic` class.

Usage

```
## S4 method for signature 'dmbc_ic'
show(object)
```

Arguments

`object`	An object of class `dmbc_ic`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`show,dmbc_model-method` *Show an instance of the* `dmbc_model` *class.*

Description

Show an instance of the `dmbc_model` class.

Usage

```
## S4 method for signature 'dmbc_model'
show(object)
```

Arguments

`object`	An object of class `dmbc_model`.

Author(s)

Sergio Venturini `<sergio.venturini@unicatt.it>`

`simdiss` *Simulated binary dissimilarity matrices.*

Description

A dataset containing a list of simulated binary dissimilarity matrices.

Usage

```
data(simdiss)
```

Format

A `dmbc_data` object whose `diss` element is a list of 10 binary dissimilarity matrices. Each matrix is defined as a `dist` object measuring the agreement among 16 different units.

Examples

```
data(simdiss)
library(bayesplot)
cols <- color_scheme_set("brightblue")
plot(simdiss, colors = unlist(cols)[c(1, 6)], font = 1, cex.font = 0.75)
```

`subset,dmbc_fit-method` *Subsetting a* `dmbc_fit` *object.*

Description

Subsetting a `dmbc_fit` object.

Usage

```
## S4 method for signature 'dmbc_fit'
subset(x, pars = character(), regex_pars = character(), ...)
```

Arguments

`x`	An object of class `dmbc_fit`.
`pars`	An optional character vector of parameter names. If neither `pars` nor `regex_pars` is specified, the default is to use all parameters.
`regex_pars`	An optional `regular expression` to use for parameter selection. Can be specified instead of `pars` or in addition to `pars`.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`subset,dmbc_fit_list-method` *Subsetting a* `dmbc_fit_list` *object.*

Description

Subsetting a `dmbc_fit_list` object.

Usage

```
## S4 method for signature 'dmbc_fit_list'
subset(x, pars = character(), regex_pars = character(), ...)
```

Arguments

`x`	An object of class `dmbc_fit_list`.
`pars`	An optional character vector of parameter names. If neither `pars` nor `regex_pars` is specified, the default is to use all parameters.
`regex_pars`	An optional `regular expression` to use for parameter selection. Can be specified instead of `pars` or in addition to `pars`.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`summary,dmbc_config-method`

Provide a summary of a `dmbc_config` *class instance.*

Description

Provide a summary of a `dmbc_config` class instance.

Usage

```
## S4 method for signature 'dmbc_config'
summary(object)
```

Arguments

`object`	An object of class `dmbc_config`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`summary,dmbc_data-method`

Provide a summary of a `dmbc_data` *class instance.*

Description

Provide a summary of a `dmbc_data` class instance.

Usage

```
## S4 method for signature 'dmbc_data'
summary(object)
```

Arguments

`object`	An object of class `dmbc_data`.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`summary,dmbc_fit-method`

Provide a summary of a `dmbc_fit` *class instance.*

Description

Provide a summary of a `dmbc_fit` class instance.

Usage

```
## S4 method for signature 'dmbc_fit'
summary(object, include.burnin = FALSE, summary.Z = FALSE, ...)
```

Arguments

`object`	An object of class `dmbc_fit`.
`include.burnin`	A length-one logical vector. If `TRUE` the burnin iterations (if available) are included in the summary.
`summary.Z`	A length-one logical vector. If `TRUE` the summary also includes the latent configuration coordinates.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`summary,dmbc_fit_list-method`

Provide a summary of a `dmbc_fit_list` *class instance.*

Description

Provide a summary of a `dmbc_fit_list` class instance.

Usage

```
## S4 method for signature 'dmbc_fit_list'
summary(object, include.burnin = FALSE, summary.Z = FALSE, ...)
```

Arguments

`object`	An object of class `dmbc_fit_list`.
`include.burnin`	A length-one logical vector. If `TRUE` the burnin iterations (if available) are included in the summary.
`summary.Z`	A length-one logical vector. If `TRUE` the summary also includes the latent configuration coordinates.
`...`	Further arguments to pass on (currently ignored).

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`summary,dmbc_ic-method` *Provide a summary of a* `dmbc_ic` *class instance.*

Description

Provide a summary of a `dmbc_ic` class instance.

Usage

```
## S4 method for signature 'dmbc_ic'
summary(object, p = NULL, G = NULL)
```

Arguments

`object`	An object of class `dmbc_ic`.
`p`	An optional length-one numeric vector providing the number of latent space dimension to use in the summary.
`G`	An optional length-one numeric vector providing the number of clusters to use in the summary.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

`update,dmbc_ic-method` *Provide an update of a* `dmbc_ic` *class instance.*

Description

Provide an update of a `dmbc_ic` class instance.

Usage

```
## S4 method for signature 'dmbc_ic'
update(object, pmax = NULL, Gmax = NULL, ...)
```

Arguments

`object`	An object of class `dmbc_ic`.
`pmax`	A length-one numeric vector indicating the maximum number of dimensions of the latent space to consider.
`Gmax`	A length-one numeric vector indicating the maximum number of cluster to consider.
`...`	Further arguments to pass on (currently ignored).

Value

A `dmbc_ic` object.

Author(s)

Sergio Venturini <`sergio.venturini@unicatt.it`>

References

Venturini, S., Piccarreta, R. (2021), "A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: the **dmbc** Package in `R`", Journal of Statistical Software, 100, 16, 1–35, <10.18637/jss.v100.i16>.

See Also

`dmbc()` for fitting a DMBC model.

`dmbc_ic` for a description of the elements included in the returned object.

Examples

```
## Not run:
data(simdiss, package = "dmbc")

pmax <- 2
Gmax <- 2
prm.prop <- list(z = 1.5, alpha = .75)
burnin <- 2000
nsim <- 1000
seed <- 1809

set.seed(seed)

control <- list(burnin = burnin, nsim = nsim, z.prop = prm.prop[["z"]],
  alpha.prop = prm.prop[["alpha"]], random.start = TRUE, verbose = TRUE,
  thin = 10, store.burnin = TRUE)
sim.ic <- dmbc_IC(data = simdiss, pmax = pmax, Gmax = Gmax, control = control,
  est = "mean")

pmax <- pmax + 1
Gmax <- Gmax + 2
new.ic <- update(sim.ic, pmax = pmax, Gmax = Gmax)
```

```
new.ic

# plot the results
library(bayesplot)
library(ggplot2)
color_scheme_set("mix-yellow-blue")
p <- plot(new.ic, size = c(4, 1.5))
p + panel_bg(fill = "gray90", color = NA)

## End(Not run)
```

Index