

# Package ‘IMIFA’

July 7, 2017

**Type** Package

**Date** 2017-07-07

**Title** Fitting, Diagnostics, and Plotting Functions for Infinite Mixtures of Infinite Factor Analysers and Related Models

**Version** 1.3.1

## Description

Provides flexible Bayesian estimation of Infinite Mixtures of Infinite Factor Analysers and related models, for nonparametrically clustering high-dimensional data, introduced by Murphy et al. (2017) <arXiv:1701.07010>. The IMIFA model conducts Bayesian nonparametric model-based clustering with factor analytic covariance structures without recourse to model selection criteria to choose the number of clusters or cluster-specific latent factors, mostly via efficient Gibbs updates. Model-specific diagnostic tools are also provided, as well as many options for plotting results and conducting posterior inference on parameters of interest.

**Depends** R (>= 3.3.2)

**License** GPL (>= 2)

**Encoding** UTF-8

**URL** <https://cran.r-project.org/package=IMIFA>

**BugReports** <https://github.com/Keefe-Murphy/IMIFA>

**Imports** abind, e1071, graphics, grDevices, matrixStats, mclust, mvnfast, plotrix, Rfast, slam, stats, utils, viridis

**Suggests** Rmpfr, gmp, knitr, methods, rmarkdown

**RoxygenNote** 6.0.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Keefe Murphy [aut, cre],  
Isobel Claire Gormley [ctb],  
Cinzia Viroli [ctb]

**Maintainer** Keefe Murphy <keefe.murphy@ucd.ie>

**Repository** CRAN

**Date/Publication** 2017-07-07 18:04:28 UTC

## R topics documented:

coffee . . . . .	2
get_IMIFA_results . . . . .	3
gumbel_max . . . . .	6
G_expected . . . . .	7
G_priorDensity . . . . .	8
G_variance . . . . .	10
heat_legend . . . . .	11
IMIFA . . . . .	11
is.cols . . . . .	13
is.posi_def . . . . .	13
Ledermann . . . . .	14
mat2cols . . . . .	15
mcmc_IMIFA . . . . .	16
MGP_check . . . . .	24
olive . . . . .	26
PGMM_dfree . . . . .	26
plot.Results_IMIFA . . . . .	28
plot_cols . . . . .	30
Procrustes . . . . .	32
psi_hyper . . . . .	33
rDirichlet . . . . .	34
shift_GA . . . . .	35
sim_IMIFA_data . . . . .	36
Zsimilarity . . . . .	37
<b>Index</b>	<b>39</b>

---

coffee

*Chemical composition of Arabica and Robusta coffee samples*

---

### Description

Data on the chemical composition of coffee samples collected from around the world, comprising 43 samples from 29 countries. Each sample is either of the Arabica or Robusta variety. Twelve of the thirteen chemical constituents reported in the study are given. The omitted variable is total chlorogenic acid; it is generally the sum of the chlorogenic, neochlorogenic and isochlorogenic acid values.

### Usage

```
data(coffee)
```

### Format

A data frame with 43 observations and 14 columns. The first two columns contain Variety (either Arabica or Robusta) and Country, respectively, while the remaining 12 columns contain the chemical properties.

## References

Streuli, H. (1973). Der heutige stand der kaffeechemie, *Association Scientifique Internationale du Cafe, 6th International Colloquium on Coffee Chemistry*, Bogata, Columbia, pp. 61-72.

---

get_IMIFA_results	<i>Extract results, conduct posterior inference and compute performance metrics for MCMC samples of models from the IMIFA family</i>
-------------------	--

---

## Description

This function post-processes simulations generated by `mcmc_IMIFA` for any of the IMIFA family of models. It can be re-ran at little computational cost in order to extract different models explored by the sampler used for sims, without having to re-run the model itself. New results objects using different numbers of clusters and different numbers of factors (if visited by the model in question), or using different model selection criteria (if necessary) can be generated with ease. The function also performs post-hoc corrections for label switching, as well as post-hoc Procrustes rotation of loadings matrices and scores, to ensure sensible posterior parameter estimates, and constructs credible intervals.

## Usage

```
get_IMIFA_results(sims = NULL, burnin = 0L, thinning = 1L, G = NULL,
  Q = NULL, criterion = c("bicm", "aicm", "log.iLLH", "dic", "bic.mcmc",
    "aic.mcmc"), G.meth = c("mode", "median"), Q.meth = c("mode", "median"),
  dat = NULL, conf.level = 0.95, z.avgsim = FALSE, z.labels = NULL)
```

## Arguments

sims	An object of class "IMIFA" generated by <code>mcmc_IMIFA</code> .
burnin	Optional additional number of iterations to discard. Defaults to 0, corresponding to no burnin.
thinning	Optional interval for extra thinning to be applied. Defaults to 1, corresponding to no thinning.
G	If this argument is not specified, results will be returned with the optimal number of clusters. If different numbers of clusters were explored in sims for the "MFA" or "MIFA" methods, supplying an integer value allows pulling out a specific solution with G clusters, even if the solution is sub-optimal. Similarly, this allows retrieval of samples corresponding to a solution, if visited, with G clusters for the "OMFA", "OMIFA", "IMFA" and "IMIFA" methods.
Q	If this argument is non specified, results will be returned with the optimal number of factors. If different numbers of factors were explored in sims for the "FA", "MFA", "OMFA" or "IMFA" methods, this allows pulling out a specific solution with Q factors, even if the solution is sub-optimal. Similarly, this allows retrieval of samples corresponding to a solution, if visited, with Q factors for the "IFA", "MIFA", "OMIFA" and "IMIFA" methods.

criterion	The criterion to use for model selection, where model selection is only required if more than one model was run under the "FA", "MFA", "MIFA", "OMFA" or "IMFA" methods when <code>sims</code> was created via <code>mcmc_IMIFA</code> . Note that these are <i>all</i> calculated, this argument merely indicates which one will form the basis of the construction of the output. Note that the first three options here might exhibit bias in favour of zero-factor models for the finite factor "FA", "MFA", "OMFA" and "IMFA" methods and might exhibit bias in favour of one-cluster models for the "MFA" and "MIFA" methods.
G.meth	If the object in <code>sims</code> arises from the "OMFA", "OMIFA", "IMFA" or "IMIFA" methods, this argument determines whether the optimal number of clusters is given by the mode or median of the posterior distribution of G. Defaults to "Mode".
Q.meth	If the object in <code>sims</code> arises from the "IFA", "MIFA", "OMIFA" or "IMIFA" methods, this argument determines whether the optimal number of latent factors is given by the mode or median of the posterior distribution of Q. Defaults to "Mode".
dat	The actual data set on which <code>mcmc_IMIFA</code> was originally run. This is necessary for computing error metrics between the estimated and empirical covariance matrix/matrices. If this is not supplied, the function will attempt to find the data set if it is still available in the global environment.
conf.level	The confidence level to be used throughout for credible intervals for all parameters of inferential interest. Defaults to 0.95.
z.avgsim	Logical indicating whether the clustering should also be summarised with a call to <code>Zsimilarity</code> by the clustering with minimum squared distance to the similarity matrix obtained by averaging the stored adjacency matrices, in addition to the MAP estimate. Note that the MAP clustering is computed <i>conditional</i> on the estimate of the number of clusters (whether that be the modal estimate or the estimate according to <code>criterion</code> ) and other parameters are extracted conditional on this estimate of G: however, in contrast, the number of distinct clusters in the summarised labels obtained by <code>z.avgsim=TRUE</code> may not necessarily coincide with the estimate of G, but may provide a useful alternative summary of the partitions explored during the chain. Please be warned that this can take considerable time to compute, and may not even be possible if the number of observations &/or number of stored iterations is large and the resulting matrix isn't sufficiently sparse, so the default is FALSE, otherwise both the summarised clustering and the similarity matrix are stored: the latter can be passed to <code>plot.Results_IMIFA</code> .
zlabels	For any method that performs clustering, the true labels can be supplied if they are known in order to compute clustering performance metrics. This also has the effect of ordering the MAP labels (and thus the ordering of cluster-specific parameters) to most closely correspond to the true labels if supplied.

### Value

An object of class "Results\_IMIFA" to be passed to `plot.Results_IMIFA` for visualising results. Dedicated print and summary functions exist for objects of this class. The object, say `x`, is a list of lists, the most important components of which are:

**Clust** Everything pertaining to clustering performance can be found here for all but the "FA" and "IFA" methods, in particular `x$Clust$map`, the MAP summary of the posterior clustering.

More detail is given if known `zlabels` are supplied: performance is always evaluated against the MAP clustering, with additional evaluation against the alternative clustering computed if `z.avgsim=TRUE`.

**Error** Error metrics (e.g. MSE) between the empirical and estimated covariance matrix/matrices.

**GQ.results** Everything pertaining to model choice can be found here, incl. posterior summaries for the estimated number of clusters and estimated number of factors, if applicable to the method employed. Information criterion values are also accessible here.

**Means** Posterior summaries for the means.

**Loadings** Posterior summaries for the factor loadings matrix/matrices. Posterior mean loadings given by `x$Loadings$post.load` are given the [loadings](#) class for printing purposes and thus the manner in which they are displayed can be modified.

**Scores** Posterior summaries for the latent factor scores.

**Uniquenesses** Posterior summaries for the uniquenesses.

## Author(s)

Keefe Murphy

## References

Murphy, K., Gormley, I. C. and Viroli, C. (2017) Infinite Mixtures of Infinite Factor Analysers: Nonparametric Model-Based Clustering via Latent Gaussian Models, [arXiv:1701.07010](#).

## See Also

[mcmc\\_IMIFA](#), [plot.Results\\_IMIFA](#), [Procrustes](#), [Zsimilarity](#)

## Examples

```
# data(coffee)
# data(olive)

# Run a MFA model on the coffee data over a range of clusters and factors.
# simMFACoffee <- mcmc_IMIFA(coffee, method="MFA", range.G=2:3, range.Q=0:3, n.iters=10000)

# Accept all defaults to extract the optimal model.
# resMFACoffee <- get_IMIFA_results(simMFACoffee)

# Instead let's get results for a 3-cluster model, allowing Q be chosen by aic.mcmc.
# resMFACoffee2 <- get_IMIFA_results(simMFACoffee, G=3, criterion="aic.mcmc")

# Run an IMIFA model on the olive data, accepting all defaults.
# simIMIFAolive <- mcmc_IMIFA(olive, method="IMIFA", n.iters=10000)

# Extract optimum results
# Estimate G & Q by the median of their posterior distributions
# Construct 90% credible intervals and try to return the similarity matrix.
# resIMIFAolive <- get_IMIFA_results(simIMIFAolive, G.meth="median", Q.meth="median",
```

```
# conf.level=0.9, z.avgsim=TRUE)
# summary(resIMIFAolive)
```

---

gumbel_max	<i>Simulate Cluster Labels from Unnormalised Log-Probabilities using the Gumbel-Max Trick</i>
------------	---

---

## Description

Samples cluster labels for  $N$  observations from  $G$  groups efficiently using log-probabilities and the so-called Gumbel-Max trick, without requiring that the log-probabilities be normalised; thus redundant computation can be avoided. Computation takes place on the log scale for stability/underflow reasons (to ensure negligible probabilities won't cause computational difficulties); in any case, many functions for calculating multivariate normal densities already output on the log scale.

## Usage

```
gumbel_max(probs, slice = FALSE)
```

## Arguments

probs	An $N \times G$ matrix of unnormalised probabilities on the log scale, where $N$ is the number of observations that require labels to be sampled and $G$ is the number of active clusters s.t. sampled labels can take values in $1:G$ .
slice	A logical indicating whether or not the indicator correction for slice sampling has been applied to probs. Defaults to FALSE but is TRUE for the "IMIFA" and "IMFA" methods under <a href="#">mcmc_IMIFA</a> . Details of this correction are given in Murphy et. al. (2017). When set to TRUE, this results in a speed-improvement when probs contains non-finite values (e.g. $-\text{Inf}$ , corresponding to zero on the probability scale).

## Value

A vector of  $N$  sampled cluster labels, with the largest label no greater than  $G$ .

## Note

Though the function is available for standalone use, note that no checks take place, in order to speed up repeated calls to the function inside [mcmc\\_IMIFA](#).

If the normalising constant is required for another reason, e.g. to compute the log-likelihood, it can be calculated by summing the output obtained by calling [rowLogSumExps](#) on probs.

## Author(s)

Keefe Murphy

## References

Murphy, K., Gormley, I. C. and Viroli, C. (2017) Infinite Mixtures of Infinite Factor Analysers: Nonparametric Model-Based Clustering via Latent Gaussian Models, [arXiv:1701.07010](#).

Yellot, J. I. Jr. (1977) The relationship between Luce's choice axiom, Thurstone's theory of comparative judgment, and the double exponential distribution, *Journal of Mathematical Psychology*, 15: 109-144.

## See Also

[mcmc\\_IMIFA](#), [rowLogSumExps](#)

## Examples

```
# Create a 1-row matrix of weights
G      <- 3
weights <- matrix(c(1, 2, 3), nrow=1, ncol=G)

# Call gumbel_max() repeatedly to obtain samples of the labels, zs
iters   <- 10000
zs      <- vapply(seq_len(iters), function(i)
  gumbel_max(probs=log(weights)), numeric(1L))

# Compare answer to the normalised weights
tabulate(zs, nbins=G)/iters
normalised <- as.numeric(weights/sum(weights))

# Simulate a matrix of dirichlet weights & the associated vector of N labels
N      <- 400
G      <- 8
sizes  <- seq(from=85, to=15, by=-10)
weights <- matrix(rDirichlet(N * G, alpha=1, nn=sizes), byrow=TRUE, nrow=N, ncol=G)
zs     <- gumbel_max(probs=log(weights))
```

---

G\_expected

---

*1st Moment of the Dirichlet / Pitman-Yor processes*


---

## Description

Calculates the expected number of clusters under a Dirichlet process or Pitman-Yor process prior for a sample of size N at given values of the concentration parameter alpha and optionally also the discount parameter. Useful for soliciting sensible priors for alpha or suitable fixed values for alpha or discount under the "IMFA" and "IMIFA" methods for [mcmc\\_IMIFA](#). All arguments are vectorised.

## Usage

```
G_expected(N, alpha, discount = 0L)
```

**Arguments**

N	The sample size.
alpha	The concentration parameter. Must be specified and must be strictly greater than -discount.
discount	The discount parameter for the Pitman-Yor process. Must lie in the interval [0, 1). Defaults to 0 (i.e. the Dirichlet process).

**Value**

The expected number of clusters under the specified prior conditions.

**Note**

Requires use of the Rmpfr and gmp libraries for non-zero discount values.

**Author(s)**

Keefe Murphy

**See Also**

[G\\_variance](#), [G\\_priorDensity](#), [Rmpfr](#)

**Examples**

```
G_expected(N=50, alpha=19.23356)

# require("Rmpfr")
# G_expected(N=50, alpha=c(19.23356, 12.21619, 1), discount=c(0, 0.25, 0.7300045))
```

---

G\_priorDensity

---

*Plot Dirichlet / Pitman-Yor process Priors*


---

**Description**

Plots the prior distribution of the number of clusters under a Dirichlet / Pitman-Yor process prior, for a sample of size N at given values of the concentration parameter alpha and optionally also the discount parameter. Useful for soliciting sensible priors for alpha or suitable fixed values for alpha or discount under the "IMFA" and "IMIFA" methods for [mcmc\\_IMIFA](#). All arguments are vectorised. Users can also consult [G\\_expected](#) and [G\\_variance](#) in order to solicit sensible priors.

**Usage**

```
G_priorDensity(N, alpha, discount = 0L, show.plot = TRUE)
```



**Arguments**

N	The sample size.
alpha	The concentration parameter. Must be specified and must be strictly greater than -discount.
discount	The discount parameter for the Pitman-Yor process. Must lie in the interval [0, 1). Defaults to 0 (i.e. the Dirichlet process).
show.plot	Logical indicating whether the plot should be displayed (default = TRUE).

**Value**

A plot of the prior distribution if `show.plot` is TRUE. Density values are returned invisibly. Note that the density values may not strictly sum to one in certain cases, as values small enough to be represented as zero may well be returned.

**Note**

Requires use of the `Rmpfr` and `gmp` libraries; may encounter difficulty and slowness for large N, especially with non-zero discount values.

**Author(s)**

Keefe Murphy

**See Also**

[G\\_expected](#), [G\\_variance](#), [Rmpfr](#)

**Examples**

```
# Plot Dirichlet process priors for different values of alpha
DP <- G_priorDensity(N=50, alpha=c(3, 10, 25))
DP

# Non-zero discount requires loading the "Rmpfr" library
# require("Rmpfr")

# Verify that these alpha/discount values produce Pitman-Yor process priors with the same mean
# G_expected(N=50, alpha=c(19.23356, 6.47006, 1), discount=c(0, 0.47002, 0.7300045))

# Now plot them to examine tail behaviour as discount increases
# PY <- G_priorDensity(N=50, alpha=c(19.23356, 6.47006, 1), discount=c(0, 0.47002, 0.7300045))
# PY
```

G\_variance

*2nd Moment of Dirichlet / Pitman-Yor processes***Description**

Calculates the variance in the number of clusters under a Dirichlet process or Pitman-Yor process prior for a sample of size N at given values of the concentration parameter alpha and optionally also the discount parameter. Useful for soliciting sensible priors for alpha or suitable fixed values for alpha or discount under the "IMFA" and "IMIFA" methods for [mcmc\\_IMIFA](#). All arguments are vectorised.

**Usage**

```
G_variance(N, alpha, discount = 0L)
```

**Arguments**

N	The sample size.
alpha	The concentration parameter. Must be specified and must be strictly greater than -discount.
discount	The discount parameter for the Pitman-Yor process. Must lie in the interval [0, 1). Defaults to 0 (i.e. the Dirichlet process).

**Value**

The variance of the number of clusters under the specified prior conditions.

**Note**

Requires use of the Rmpfr and gmp libraries for non-zero discount values.

**Author(s)**

Keefe Murphy

**See Also**

[G\\_expected](#), [G\\_priorDensity](#), [Rmpfr](#)

**Examples**

```
G_variance(N=50, alpha=19.23356)

# require("Rmpfr")
# G_variance(N=50, alpha=c(19.23356, 12.21619, 1), discount=c(0, 0.25, 0.7300045))
```

---

heat_legend	<i>Add a colour key legend to heatmap plots</i>
-------------	---

---

**Description**

Using only base graphics, this function appends a colour key legend for heatmaps produced by, for instance, `plot_cols` or `image`.

**Usage**

```
heat_legend(data, cols)
```

**Arguments**

data	Either the data with which the heatmap was created or a vector containing its minimum and maximum values. Missing values are ignored.
cols	The palette used when the heatmap was created.

**Value**

Modifies an existing plot by adding a legend.

**See Also**

`image`, `plot_cols`, `mat2cols`, `is.cols`

**Examples**

```
# Generate a matrix, flip it, and plot it with a legend
data <- matrix(rnorm(50), nrow=10, ncol=5)
cols <- heat.colors(12)[12:1]
par(mar=c(5.1, 4.1, 4.1, 4.1))

image(t(data)[,nrow(data):1], col=cols)
heat_legend(data, cols); box(lwd=2)
```

---

IMIFA	<i>IMIFA: Fitting, Diagnostics, and Plotting Functions for Infinite Mixtures of Infinite Factor Analysers and Related Models</i>
-------	--

---

**Description**

A package for Bayesian nonparameteric clustering of high-dimensional datasets, providing functions for fitting, diagnostic tools and plotting for Infinite Mixtures of Infinite Factor Analysers and the full suite of related models. Allows model based clustering with factor analytic covariance structures without recourse to model selection criteria to choose the number of clusters or cluster-specific latent factors. Model-specific diagnostic tools are also provided, as well as many options for plotting results and posterior distributions of parameters of inferential interest.

## Usage

The three most important functions in the **IMIFA** package are: `mcmc_IMIFA`, for fitting the model, `get_IMIFA_results`, for extracting results from objects of the "IMIFA" class generated by `mcmc_IMIFA`, and the dedicated plotting function `plot.Results_IMIFA`, for plotting results pertaining to parameters of inferential interest from objects of class "Results\_IMIFA" generated by `get_IMIFA_results`.

Other functions also exist for simulating data from a multivariate mixture of factor analysers and for soliciting good priors.

## Details

Package: IMIFA

Type: Package

Version: 1.3.1

Date: 2017-07-07 (this version), 2017-02-02 (original release)

Licence: GPL (>=2)

`mcmc_IMIFA`: This function estimates models in the IMIFA family under the Bayesian paradigm. Most importantly, one must specify the 'method' in the form of an acronym (e.g. "MIFA" for Mixtures of Infinite Factor Analysers) and ranges of values for range.G, the number of clusters, and range.Q, the number(s) of (cluster-specific) latent factors as required by said method.

`get_IMIFA_results`: Raw simulation objects generated by `mcmc_IMIFA()` are passed to this function in order to extract results of interest and conduct further post-processing if necessary.

`plot.Results_IMIFA`: Results obtained from `get_IMIFA_Results` are passed to this function with the type of plot desired specified by 'plot.meth' (e.g. "trace") and the parameter of interest specified by 'param' (e.g. "loadings").

## References

Murphy, K., Gormley, I. C. and Viroli, C. (2017) Infinite Mixtures of Infinite Factor Analysers: Nonparametric Model-Based Clustering via Latent Gaussian Models, [arXiv:1701.07010](#)

## See Also

Further details and examples are given in the associated vignette document:  
`vignette("IMIFA", package = "IMIFA")`

## Author(s)

Keefe Murphy [aut, cre], Isobel Claire Gormley [ctb], Cinzia Viroli [ctb]

Maintainer: Keefe Murphy - [<keefe.murphy@ucd.ie>](mailto:keefe.murphy@ucd.ie)

---

is.cols	<i>Check for Valid Colours</i>
---------	--------------------------------

---

**Description**

Checks if the supplied vector contains valid colours.

**Usage**

```
is.cols(cols)
```

**Arguments**

cols	A vector of colours, usually as a character string.
------	---

**Value**

A logical vector of length `length(cols)` which is TRUE for entries which are valid colours and FALSE otherwise.

**Examples**

```
all(is.cols(1:5))

all(is.cols(heat.colors(30)))

any(!is.cols(c("red", "green", "aquamarine")))
```

---

is.posi_def	<i>Check Postive-(Semi)definiteness of a matrix</i>
-------------	---

---

**Description**

Tests whether all eigenvalues of a symmetric matrix are positive (or strictly non-negative) to check for positive-definiteness and positive-semidefiniteness, respectively. If the supplied matrix doesn't satisfy the test, the nearest matrix which does can optionally be returned.

**Usage**

```
is.posi_def(x, tol = NULL, semi = FALSE, make = FALSE)
```

**Arguments**

<code>x</code>	A matrix, assumed to be real and symmetric.
<code>tol</code>	Tolerance for singular values and for absolute eigenvalues - only those with values larger than <code>tol</code> are considered non-zero (default: <code>tol = max(dim(x))*max(E)*.Machine\$double.eps</code> , where <code>E</code> is the vector of absolute eigenvalues).
<code>semi</code>	Logical switch to test for positive-semidefiniteness when <code>TRUE</code> or positive-definiteness when <code>FALSE</code> (the default).
<code>make</code>	Logical switch to return the nearest matrix which satisfies the test - if the test has been passed, this is of course just <code>x</code> itself, otherwise the nearest positive-(semi)definite matrix. Note that for reasons due to finite precision arithmetic, finding the nearest positive-definite and nearest positive-semidefinite matrices are effectively equivalent tasks.

**Value**

If `isTRUE(make)`, a list with two components:

**check** A logical value indicating whether the matrix satisfies the test.

**X.new** The nearest matrix which satisfies the test (which may just be the input matrix itself.)

Otherwise, only the logical value indicating whether the matrix satisfies the test is returned.

**Examples**

```
x <- cov(matrix(rnorm(100), nrow=10, ncol=10))
is.posi_def(x)
is.posi_def(x, semi=TRUE)

Xnew <- is.posi_def(x, semi=FALSE, make=TRUE)$X.new
identical(x, Xnew)
identical(x, is.posi_def(x, semi=TRUE, make=TRUE)$X.new)
```

---

Ledermann

*Ledermann Bound*


---

**Description**

Returns the maximum possible number of latent factors in a factor analysis model for data of dimension  $P$ . This Ledermann bound is given by the largest integer smaller than or equal to the solution  $k$  of  $(M - k)^2 \geq M + k$ .

**Usage**

```
Ledermann(P)
```

**Arguments**

`P` Integer number of variables in data set.

**Value**

The Ledermann bound, a non-negative integer.

**Examples**

```
Ledermann(25)
```

---

mat2cols	<i>Convert a numeric matrix to colours</i>
----------	--

---

**Description**

Converts a matrix to a hex colour code representation for plotting using [plot\\_cols](#). Used internally by [plot.Results\\_IMIFA](#) for plotting posterior mean loadings heatmaps.

**Usage**

```
mat2cols(mat, cols = NULL, compare = FALSE, byrank = FALSE,
         breaks = length(cols), na.col = "#808080FF")
```

**Arguments**

mat	Either a matrix or, when compare is TRUE, a list of matrices.
cols	The colour palette to be used. The default palette uses <a href="#">viridis</a> . Will be checked for validity.
compare	Logical switch used when desiring comparable colour representations (usually for comparable heat maps) across multiple matrices. Ensures plots will be calibrated to a common colour scale so that, for instance, the colour on the heat map of an entry valued at 0.7 in Matrix A corresponds exactly to the colour of a similar value in Matrix B. When TRUE, mat must be supplied as a list of matrices, which must have either the same number of rows, or the same number of columns.
byrank	Logical indicating whether to convert the matrix itself or the sample ranks of the values therein. Defaults to FALSE.
breaks	Number of gradations in colour to use. Defaults to length(cols).
na.col	Colour to be used to represent missing data.

**Value**

A matrix of hex colour code representations, or a list of such matrices when compare is TRUE.

**See Also**

[plot\\_cols](#), [heat\\_legend](#), [is.cols](#)

## Examples

```
# Generate a colour matrix using mat2cols()
mat      <- matrix(rnorm(100), nrow=10, ncol=10)
mat[2,3] <- NA
cols     <- heat.colors(12)[12:1]
matcol   <- mat2cols(mat, cols=cols)
matcol

# Use plot_cols() to visualise the colours matrix
par(mar=c(5.1, 4.1, 4.1, 4.1))
plot_cols(matcol)

# Add a legend using heat_legend()
heat_legend(mat, cols=cols); box(lwd=2)

# Try comparing heat maps of multiple matrices
mat1     <- matrix(rnorm(100), nr=50, nc=2)
mat2     <- matrix(rnorm(150), nr=50, nc=3)
mat3     <- matrix(rnorm(50), nr=50, nc=1)
mats     <- list(mat1, mat2, mat3)
colmats  <- mat2cols(mats, cols=cols, compare=TRUE)
par(mfrow=c(2, 3), mar=c(1, 2, 1, 2))

# Use common palettes (top row)
plot_cols(colmats[[1]]); heat_legend(range(mats), cols=cols); box(lwd=2)
plot_cols(colmats[[2]]); heat_legend(range(mats), cols=cols); box(lwd=2)
plot_cols(colmats[[3]]); heat_legend(range(mats), cols=cols); box(lwd=2)

# Use uncommon palettes (bottom row)
plot_cols(mat2cols(mat1, cols=cols)); heat_legend(range(mats), cols=cols); box(lwd=2)
plot_cols(mat2cols(mat2, cols=cols)); heat_legend(range(mats), cols=cols); box(lwd=2)
plot_cols(mat2cols(mat3, cols=cols)); heat_legend(range(mats), cols=cols); box(lwd=2)
```

---

mcmc\_IMIFA

---

*Adaptive Gibbs Sampler for Nonparameteric Model-based Clustering  
using models from the IMIFA family*


---

## Description

Carries out Gibbs sampling for all models from the IMIFA family, facilitating model-based clustering with dimensionally reduced factor-analytic covariance structures, with automatic estimation of the number of clusters and cluster-specific factors as appropriate to the method employed. Factor analysis with one group (FA/IFA), finite mixtures (MFA/MIFA), overfitted mixtures (OMFA/OMIFA), infinite factor models which employ the multiplicative gamma process (MGP) shrinkage prior (IFA/MIFA/OMIFA/IMIFA), and infinite mixtures which employ Dirichlet Process Mixture Models (IMFA/IMIFA) are all provided. Creates a raw object of class 'IMIFA' from which the optimal/modal model can be extracted by `get_IMIFA_results`.



**Usage**

```
mcmc_IMIFA(dat = NULL, method = c("IMIFA", "IMFA", "OMIFA", "OMFA", "MIFA",
  "MFA", "IFA", "FA", "classify"), n.iters = 25000L, range.G = NULL,
  range.Q = NULL, burnin = n.iters/5, thinning = 2L, centering = TRUE,
  scaling = c("unit", "pareto", "none"), uni.type = c("unconstrained",
  "isotropic", "constrained", "single"), uni.prior = c("unconstrained",
  "isotropic"), alpha = NULL, psi.alpha = NULL, psi.beta = NULL,
  mu.zero = NULL, sigma.mu = NULL, sigma.l = NULL, z.init = c("mclust",
  "kmeans", "list", "priors"), z.list = NULL, adapt = TRUE, prop = NULL,
  epsilon = NULL, alpha.d1 = NULL, alpha.d2 = NULL, beta.d1 = NULL,
  beta.d2 = NULL, nu = NULL, nuplus1 = TRUE, adapt.at = NULL,
  b0 = NULL, b1 = NULL, trunc.G = NULL, learn.alpha = TRUE,
  alpha.hyper = NULL, ind.slice = TRUE, rho = NULL, IM.lab.sw = TRUE,
  discount = NULL, learn.d = FALSE, d.hyper = NULL, kappa = NULL,
  zeta = NULL, tune.zeta = NULL, mu0g = FALSE, psi0g = FALSE,
  delta0g = FALSE, mu.switch = TRUE, score.switch = TRUE,
  load.switch = TRUE, psi.switch = TRUE, pi.switch = TRUE,
  verbose = interactive())
```

**Arguments**

<code>dat</code>	A matrix or data frame such that rows correspond to observations (N) and columns correspond to variables (P). Non-numeric variables and rows with missing entries will be removed.
<code>method</code>	An acronym for the type of model to fit where: "FA" Factor Analysis "MFA" Mixtures of Factor Analysers "MIFA" Mixtures of Infinite Factor Analysers "OMFA" Overfitted Mixtures of Factor Analysers "OMIFA" Overfitted Mixtures of Infinite Factor Analysers "IMFA" Infinite Mixtures of Factor Analysers "IMIFA" Infinite Mixtures of Infinite Factor Analysers The "classify" method is not yet implemented.
<code>n.iters</code>	The number of iterations to run the Gibbs sampler for.
<code>range.G</code>	Depending on the method employed, either the range of values for the number of clusters, or the conservatively high starting value for the number of clusters. Defaults to 1 for the "FA" and "IFA" methods. For the "MFA" and "MIFA" models this is to be given as a range of candidate models to explore. For the "OMFA", "OMIFA", "IMFA", and "IMIFA" models, this is the number of clusters with which the chain is to be initialised, in which case the default is $\min(N - 1, \max(25, \text{ceiling}(3 * \log(N))))$ . For the "OMFA", and "OMIFA" models this upper limit remains fixed for the entire length of the chain; <code>range.G</code> also doubles as the default <code>trunc.G</code> for the "IMFA" and "IMIFA" models. However, when $N < P$ , or when this bound is close to or exceeds N for any of these overfitted/infinite mixture models, it is better to initialise at a value closer to the truth (i.e. $\text{ceiling}(\log(N))$ by default), though the upper bound remains the same - as a result the role of <code>range.G</code> when $N < P$ is

no longer to specify the upper bound (which can still be modified via `trunc.G`, at least for the "IMFA" and "IMIFA" methods) and the number of groups used for initialisation, but rather just the number of groups used for initialisation only. If `length(range.G) * length(range.Q)` is large, consider not storing unnecessary parameters, or breaking up the range of models to be explored into chunks, and sending each chunk to `get_IMIFA_results`.

<code>range.Q</code>	Depending on the method employed, either the range of values for the number of latent factors, or, for methods ending in IFA the conservatively high starting value for the number of cluster-specific factors, in which case the default starting value is <code>floor(3 * log(P))</code> . For methods ending in IFA, different clusters can be modelled using different numbers of latent factors (incl. zero); for methods not ending in IFA it is possible to fit zero-factor models, corresponding to simple diagonal covariance structures. For instance, fitting the "IMFA" model with <code>range.Q=0</code> corresponds to a vanilla Dirichlet Process Mixture Model. If <code>length(range.G) * length(range.Q)</code> is large, consider not storing unnecessary parameters or breaking up the range of models to be explored into chunks, and sending each chunk to <code>get_IMIFA_results</code> .
<code>burnin</code>	The number of burn-in iterations for the sampler. Defaults to <code>n.iters/5</code> . Note that chains can also be burned in later, using <code>get_IMIFA_results</code> .
<code>thinning</code>	The thinning interval used in the simulation. Defaults to 2. No thinning corresponds to 1. Note that chains can also be thinned later, using <code>get_IMIFA_results</code> .
<code>centering</code>	A logical value indicating whether mean centering should be applied to the data, defaulting to TRUE.
<code>scaling</code>	The scaling to be applied - one of "unit", "none" or "pareto".
<code>uni.type</code>	This argument specifies the type of constraint, if any, to be placed on the uniquenesses/idiosyncratic variances, i.e. whether a general diagonal matrix or isotropic diagonal matrix is to be assumed, and in turn whether these matrices are constrained to be equal across groups. The default "unconstrained" corresponds to factor analysis (and mixtures thereof), whereas "isotropic" corresponds to probabilistic principal components analysers (and mixtures thereof). Constraints <i>may</i> be particularly useful when $N < P$ , though caution is advised when employing constraints for any of the infinite factor models, especially "isotropic" and "single", which may lead to overestimation of the number of clusters &/or factors if this specification is inappropriate. The four options correspond to the following 4 parsimonious Gaussian mixture models:

"unconstrained" **UUU** - variable-specific and cluster-specific:  $\Psi_g = \Psi_g$ .

"isotropic" **UUC** - cluster-specific, equal across variables:  $\Psi_g = \sigma_g^2 \mathcal{I}_p$ .

"constrained" **UCU** - variable-specific, equal across clusters:  $\Psi_g = \Psi$ .

"single" **UCC** - a single value equal across clusters and variables:  $\Psi_g = \sigma^2 \mathcal{I}_p$ .

The first letter **U** here corresponds to constraints on loadings (not yet implemented), the second letter corresponds to constrained/unconstrained across clusters, and the third letter corresponds to the isotropic constraint. Of course, only the third letter is of relevance for the single-cluster "FA" and "IFA" models, such that "unconstrained" and "constrained" are equivalent for these models, and so too are "isotropic" and "single".

<code>uni.prior</code>	A switch indicating whether uniquenesses rate hyperparameters are to be "unconstrained" or "isotropic", i.e. variable-specific or not. "uni.prior" must be "isotropic" if the last letter of "uni.type" is C, but can take either value otherwise. Defaults to correspond to the last letter of uni.type if that is supplied and uni.prior is not, otherwise defaults to "unconstrained", but "isotropic" is recommended when $N < P$ . Only relevant when "psi.beta" is not supplied and <code>psi_hyper</code> is therefore invoked.
<code>alpha</code>	Depending on the method employed, either the hyperparameter of the Dirichlet prior for the cluster mixing proportions, or the Dirichlet process concentration parameter. Defaults to 0.5/range.G for the Overfitted methods - if supplied for "OMFA" and "OMIFA" methods, you are supplying the numerator of alpha/range.G, which should be less than half the dimension (per group!) of the free parameters of the smallest model considered in order to ensure superfluous clusters are emptied (for "OMFA", this corresponds to the smallest range.Q; for "OMIFA", this corresponds to a zero-factor model) [see: <code>PGMM_dfree</code> and Rousseau and Mengersen (2011)]. Defaults to 1 for the finite mixture models "MFA" and "MIFA". Defaults to 1 - discount for the "IMFA" and "IMIFA" models if <code>learn.alpha=FALSE</code> or a simulation from the prior if <code>learn.alpha=TRUE</code> . Must be positive, unless discount is supplied for the "IMFA" or "IMIFA" methods.
<code>psi.alpha</code>	The shape of the inverse gamma prior on the uniquenesses. Defaults to 2.5 if uni.type is one of "unconstrained" or "constrained", otherwise defaults to 3.5.
<code>psi.beta</code>	The rate of the inverse gamma prior on the uniquenesses. Can be either a single parameter, a vector of variable specific rates, or a matrix of variable and group-specific rates. If this is not supplied, <code>psi_hyper</code> is invoked to choose sensible values, depending on the value of uni.prior and, for the "MFA" and "MIFA" models, the value of <code>psi0g</code> .
<code>mu.zero</code>	The mean of the prior distribution for the mean parameter. Defaults to the sample mean of the data.
<code>sigma.mu</code>	The covariance of the prior distribution for the mean parameter. Can be a scalar times the identity or a matrix of appropriate dimension. Defaults to the sample covariance matrix.
<code>sigma.l</code>	The covariance of the prior distribution for the loadings. Defaults to 1. Only relevant for the finite factor methods.
<code>z.init</code>	The method used to initialise the cluster labels. Defaults to <code>Mclust</code> . Not relevant for the "FA" and ""IFA" methods.
<code>z.list</code>	A user supplied list of cluster labels. Only relevant if <code>z.init == "z.list"</code> .
<code>adapt</code>	A logical value indicating whether adaptation of the number of cluster-specific factors is to take place. Only relevant for methods ending in IFA, in which case the default is TRUE. Specifying FALSE and supplying range.Q provides a means to use the MGP prior in a finite factor context.
<code>prop</code>	Proportion of elements within the neighbourhood epsilon of zero necessary to consider a loadings column redundant. Defaults to $\text{floor}(0.7 * P)/P$ . Only relevant for methods ending in IFA.

epsilon	Neighbourhood of zero within which a loadings entry is considered negligible according to prop. Defaults to 0.1. Only relevant for methods ending in IFA.
alpha.d1	Shape hyperparameter of the global shrinkage on the first column of the loadings according to the MGP shrinkage prior. Passed to <a href="#">MGP_check</a> to ensure validity. Defaults to 3. Only relevant for methods ending in IFA.
alpha.d2	Shape hyperparameter of the global shrinkage on subsequent columns of the loadings according to the MGP shrinkage prior. Passed to <a href="#">MGP_check</a> to ensure validity. Defaults to 6. Only relevant for methods ending in IFA.
beta.d1	Rate hyperparameter of the global shrinkage on the first column of the loadings according to the MGP shrinkage prior. Passed to <a href="#">MGP_check</a> to ensure validity. Defaults to 1. Only relevant for methods ending in IFA.
beta.d2	Rate hyperparameter of the global shrinkage on the first column of the loadings according to the MGP shrinkage prior. Passed to <a href="#">MGP_check</a> to ensure validity. Defaults to 1. Only relevant for methods ending in IFA.
nu	Hyperparameter for the gamma prior on the local shrinkage parameters. Defaults to 2. Passed to <a href="#">MGP_check</a> to ensure validity. Only relevant for methods ending in IFA.
nuplus1	Logical switch indicating whether the shape hyperparameter of the prior on the local shrinkage parameters is equal to $\nu + 1$ . If FALSE, it is simply equal to $\nu$ . Only relevant for methods ending in IFA.
adapt.at	The iteration at which adaptation is to begin. Defaults to burnin for the "IFA" and "MIFA" methods, defaults to 0 for the "OMIFA" and "IMIFA". Cannot exceed burnin. Only relevant for methods ending in IFA.
b0	Intercept parameter for the exponentially decaying adaptation probability s.t. $p(\text{iter}) = 1/\exp(b0 + b1 * (\text{iter} - \text{adapt.at}))$ . Defaults to 0.1. Only relevant for methods ending in IFA.
b1	Slope parameter for the exponentially decaying adaptation probability s.t. $p(\text{iter}) = 1/\exp(b0 + b1 * (\text{iter} - \text{adapt.at}))$ . Defaults to 0.00005. Only relevant for methods ending in IFA.
trunc.G	The maximum number of allowable and storable groups if the "IMFA" or "IMIFA" method is employed. Defaults to the same value as range.G (unless $N < P$ , see range.G for details) and must be greater than or equal to this value. The number of active groups to be sampled at each iteration is adaptively truncated, with trunc.G as an upper limit for storage reasons. Note that large values of trunc.G may lead to memory capacity issues.
learn.alpha	Logical indicating whether the Dirichlet process / Pitman concentration parameter is to be learned, or remain fixed for the duration of the chain. If being learned, a $Ga(a, b)$ prior is assumed for alpha; updates take place via Gibbs sampling when discount is zero and via Metropolis-Hastings otherwise. Only relevant for the "IMFA" and "IMIFA" methods, in which case the default is TRUE.
alpha.hyper	A vector of length 2 giving hyperparameters for the Dirichlet process / Pitman-Yor concentration parameter alpha. If <code>isTRUE(learn.alpha)</code> , these are shape and rate parameter of a Gamma distribution. Defaults to $Ga(2, 1)$ . Only relevant for the "IMFA" and "IMIFA" methods, in which case the default is TRUE. The prior is shifted to have support on $(-\text{discount}, \text{Inf})$ when non-zero discount is supplied or <code>learn.d=TRUE</code> .

<code>ind.slice</code>	Logical indicating whether the independent slice-efficient sampler is to be employed. If FALSE the dependent slice-efficient sampler is employed, whereby the slice sequence $\mathbf{x}_1, \dots, \mathbf{x}_g$ is equal to the decreasingly ordered mixing proportions. Only relevant for the "IMFA" and "IMIFA" methods. Defaults to TRUE.
<code>rho</code>	Parameter controlling the rate of geometric decay for the independent slice-efficient sampler, s.t. $\mathbf{x}_i = (1 - \rho)\rho^{(g-1)}$ . Must lie in the interval (0, 1]. Higher values are associated with better mixing but longer run times. Defaults to 0.75, but 0.5 is an interesting special case which guarantees that the slice sequence $\mathbf{x}_1, \dots, \mathbf{x}_g$ is equal to the <i>expectation</i> of the decreasingly ordered mixing proportions. Only relevant for the "IMFA" and "IMIFA" methods when <code>ind.slice</code> is TRUE.
<code>IM.lab.sw</code>	Logical indicating whether the two forced label switching moves are to be implemented (defaults to TRUE) when running one of the infinite mixture models, with Dirichlet process or Pitman-Yor process priors. Only relevant for the "IMFA" and "IMIFA" methods.
<code>discount</code>	The discount parameter used when generalising the Dirichlet process to the Pitman-Yor process. Must lie in the interval [0, 1). If non-zero, $\alpha$ can be supplied greater than $-\text{discount}$ . Defaults to 0. Only relevant for the "IMFA" and "IMIFA" methods.
<code>learn.d</code>	Logical indicating whether the <code>discount</code> parameter is to be updated via Metropolis-Hastings. Only relevant for the "IMFA" and "IMIFA" methods, in which case the default is FALSE.
<code>d.hyper</code>	Hyperparameters for the Beta(a,b) prior on the discount hyperparameter. Only relevant for the "IMFA" and "IMIFA" methods.
<code>kappa</code>	The spike-and-slab prior distribution on the discount hyperparameter is assumed to be a mixture with point-mass at zero and a continuous Beta(a,b) distribution. <code>kappa</code> gives the weight of the point mass at zero (the 'spike'). Must lie in the interval [0,1]. Defaults to 0.5. Only relevant for the "IMFA" and "IMIFA" methods when <code>isTRUE(learn.d)</code> . A value of 0 ensures non-zero discount values (i.e. Pitman-Yor) at all times, and <i>vice versa</i> .
<code>zeta</code>	Tuning parameter controlling the acceptance rate of the random-walk proposal for the Metropolis-Hastings steps when <code>learn.alpha=TRUE</code> , where $2 * \text{zeta}$ gives the full width of the uniform proposal distribution. These steps are only invoked when either <code>discount</code> is non-zero and fixed or <code>learn.d=TRUE</code> , otherwise $\alpha$ is learned by Gibbs updates. Must be strictly positive. Defaults to 2.
<code>tune.zeta</code>	Used for tuning <code>zeta</code> & the width of the uniform proposal for $\alpha$ via diminishing Robbins-Monro type adaptation, when that parameter is learned via Metropolis-Hastings. Must be given in the form of a list with the following <i>named</i> elements: <p>"heat" The initial adaptation intensity/step-size, such that larger values lead to larger updates. Must be strictly greater than zero. Defaults to 1 if not supplied but other elements of <code>tune.zeta</code> are.</p> <p>"lambda" Iteration rescaling parameter which controls the speed at which adaptation diminishes, such that lower values cause the contribution of later iterations to diminish more slowly. Must lie in the interval (0.5, 1]. Defaults to 1 if not supplied but other elements of <code>tune.zeta</code> are.</p>

"target" The target acceptance rate. Must lie in the interval [0, 1]. Defaults to 0.441, which is optimum for univariate targets, if not supplied but other elements of `tune.zeta` are.

`tune.zeta` is only relevant when `isTRUE(learn.alpha)` under the "IMFA" or "IMIFA" models, and either the discount remains fixed at a non-zero value, or when `isTRUE(learn.d)` and `kappa < 1`. Since Gibbs steps are invoked for updated alpha when `discount == 0`, adaption occurs according to a running count of the number of iterations with non-zero sampled discount values. If diminishing adaptation invoked, the posterior mean zeta will be stored. Since caution is advised when employing adaptation, note that acceptance rates of between 10-50% are generally considered adequate.

<code>mu0g</code>	Logical indicating whether the <code>mu.zero</code> hyperparameter can be cluster-specific. Defaults to FALSE. Only relevant for the "MFA" and "MIFA" methods when <code>z.list</code> is supplied.
<code>psi0g</code>	Logical indicating whether the <code>psi.beta</code> hyperparameter(s) can be cluster-specific. Defaults to FALSE. Only relevant for the "MFA" and "MIFA" methods when <code>z.list</code> is supplied, and only allowable when <code>uni.type</code> is one of <code>unconstrained</code> or <code>isotropic</code> .
<code>delta0g</code>	Logical indicating whether the <code>alpha.d1</code> and <code>alpha.d2</code> hyperparameters can be cluster-specific. Defaults to FALSE. Only relevant for the "MIFA" method when <code>z.list</code> is supplied.
<code>mu.switch</code>	Logical indicating whether the means are to be stored (defaults to TRUE). May be useful not to store if memory is an issue. Warning: posterior inference won't be possible.
<code>score.switch</code>	Logical indicating whether the factor scores are to be stored. As the array containing each sampled scores matrix tends to be amongst the largest objects to be stored, this defaults to FALSE when <code>length(range.G) * length(range.Q) &gt; 10</code> , otherwise the default is TRUE. May be useful not to store if memory is an issue - for the "MIFA", "OMIFA", and "IMIFA" methods, setting this switch to FALSE also offers a slight speed-up. Warning: posterior inference won't be possible.
<code>load.switch</code>	Logical indicating whether the factor loadings are to be stored (defaults to TRUE). May be useful not to store if memory is an issue. Warning: posterior inference won't be possible.
<code>psi.switch</code>	Logical indicating whether the uniquenesses are to be stored (defaults to TRUE). May be useful not to store if memory is an issue. Warning: posterior inference won't be possible.
<code>pi.switch</code>	Logical indicating whether the mixing proportions are to be stored (defaults to TRUE). May be useful not to store if memory is an issue. Warning: posterior inference won't be possible.
<code>verbose</code>	Logical indicating whether to print output (e.g. run times) and a progress bar to the screen while the sampler runs. By default is TRUE if the session is interactive, and FALSE otherwise. If FALSE, warnings and error messages will still be printed to the screen, but everything else will be suppressed.

**Value**

A list of lists of lists of class "IMIFA" to be passed to [get\\_IMIFA\\_results](#). If the returned object is `x`, candidate models accessible via subsetting, where `x` is of the form `x[[1:length(range.G)]][[1:length(range.Q)]]`. However, these objects of class "IMIFA" should rarely if ever be manipulated by hand - use of the [get\\_IMIFA\\_results](#) function is *strongly* advised. Dedicated print and summary functions exist for objects of class "IMIFA".

**Author(s)**

Keefe Murphy

**References**

- Murphy, K., Gormley, I. C. and Viroli, C. (2017) Infinite Mixtures of Infinite Factor Analysers: Nonparametric Model-Based Clustering via Latent Gaussian Models, [arXiv:1701.07010](#).
- Bhattacharya, A. and Dunson, D. B. (2011) Sparse Bayesian infinite factor models, *Biometrika*, 98(2): 291-306.
- Kalli, M., Griffin, J. E. and Walker, S. G. (2011) Slice sampling mixture models, *Statistics and Computing*, 21(1): 93-105.
- McNicholas, P. D. and Murphy, T. B. (2008) Parsimonious Gaussian Mixture Models, *Statistics and Computing*, 18(3): 285-296.
- Rousseau, J. and Mengersen, K. (2011) Asymptotic Behaviour of the posterior distribution in over-fitted mixture models, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(5): 689-710.
- Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3): 611-622.

**See Also**

[get\\_IMIFA\\_results](#), [psi\\_hyper](#), [MGP\\_check](#)

**Examples**

```
# data(olive)
# data(coffee)

# Fit an IMIFA model to the olive data. Accept all defaults.
# simIMIFA <- mcmc_IMIFA(olive, method="IMIFA")
# summary(simIMIFA)

# Fit an IMIFA model assuming a Pitman-Yor prior.
# Allow the alpha and discount parameter to be learned.
# Control the balance between the DP and PY priors using the kappa parameter.
# simPY <- mcmc_IMIFA(olive, method="IMIFA", learn.d=TRUE, kappa=0.5)
# summary(simPY)

# Fit a MFA model to the scaled olive data, with isotropic uniquenesses (i.e. MPPCA).
# Allow diagonal covariance as a special case where range.Q = 0. Accept all other defaults.
# simMFA <- mcmc_IMIFA(olive, method="MFA", n.iters=10000, range.G=3:6,
```

```
#                               range.Q=0:3, centering=FALSE, uni.type="isotropic")

# Fit a MIFA model to the centered & scaled coffee data, w/ cluster labels initialised by K-Means.
# Note that range.Q doesn't need to be specified. Allow IFA as a special case where range.G=1.
# simMIFA <- mcmc_IMIFA(coffee, method="MIFA", n.iters=10000, range.G=1:3, z.init="kmeans")

# Fit an IFA model to the centered and pareto scaled olive data.
# Note that range.G doesn't need to be specified. We can optionally supply a range.Q starting value.
# We can also enforce additional shrinkage using alpha.d1, alpha.d2, prop, and epsilon.
# simIFA <- mcmc_IMIFA(olive, method="IFA", n.iters=10000, range.Q=4, scaling="pareto",
#                       alpha.d1=3.5, alpha.d2=7, prop=0.6, epsilon=0.12)

# Fit an OMIFA model to the centered & scaled coffee data.
# Supply a sufficiently small alpha value. Try varying other hyperparameters.
# Accept the default value for the starting number of factors,
# but supply a value for the starting number of clusters.
# Try constraining uniquenesses to be common across both variables and clusters.
# simOMIFA <- mcmc_IMIFA(coffee, method="OMIFA", range.G=10, psi.alpha=3,
#                        nu=3, alpha=0.8, uni.type="single")
```

MGP\_check

*Check the validity of Multiplicative Gamma Process (MGP) hyperparameters*

## Description

Checks the hyperparameters for the multiplicative gamma process (MGP) shrinkage prior in order to ensure that the property of cumulative shrinkage holds. This is called inside `mcmc_IMIFA` for the "IFA", "MIFA", "OMIFA" and "IMIFA" methods. The arguments `ad1`, `ad2`, `nu`, `bd1` and `bd2` are vectorised.

## Usage

```
MGP_check(ad1, ad2, Q, nu, bd1 = 1L, bd2 = 1L, plus1 = TRUE,
          inverse = TRUE)
```

## Arguments

<code>ad1</code>	Shape hyperparameter for $\delta_1$ .
<code>ad2</code>	Shape hyperparameter for $\delta_2$ .
<code>Q</code>	Number of latent factors.
<code>nu</code>	Hyperparameter for the local shrinkage parameters.
<code>bd1</code>	Rate hyperparameter for $\delta_1$ . Defaults to 1.
<code>bd2</code>	Rate hyperparameter for $\delta_2$ . Defaults to 1.
<code>plus1</code>	Logical indicator for whether the Gamma prior on the local shrinkage parameters is of the form $\text{Ga}(\nu + 1, \nu)$ , the default, or $\text{Ga}(\nu, \nu)$ .



**inverse** Logical indicator for whether the cumulative shrinkage property is assessed against the induced Inverse Gamma prior, the default, or in terms of the Gamma prior (which is incorrect). This is always TRUE when used inside [mcmc\\_IMIFA](#): the FALSE option exists only for demonstration purposes.

### Value

A list of length 2 containing the following objects:

**expectation** The vector of actual expected shrinkage factors, i.e. the inverse of the global shrinkage parameters.

**valid** A logical indicating whether the cumulative shrinkage property holds.

### Author(s)

Keefe Murphy

### References

Bhattacharya, A. and Dunson, D. B. (2011). Sparse Bayesian infinite factor models, *Biometrika*, 98(2): 291-306.

Durante, D. (2017). A note on the multiplicative gamma process, *Statistics & Probability Letters*, 122: 198-204.

### See Also

[mcmc\\_IMIFA](#)

### Examples

```
# Check if expected shrinkage under the MGP increases with the column index (WRONG!).
MGP_check(ad1=1.5, ad2=1.8, Q=10, nu=2, inverse=FALSE)[[1]]$valid

# Check if the induced IG prior on the MGP global shrinkage parameters
# is stochastically increasing, thereby inducing cumulative shrinkage (CORRECT!).
MGP_check(ad1=1.5, ad2=1.8, Q=10, nu=2, inverse=TRUE)[[1]]$valid

# Check again with a parameterisation that IS valid and examine the expected shrinkage values.
shrink <- MGP_check(ad1=1.5, ad2=2.8, Q=10, nu=2, inverse=TRUE)[[1]]
shrink$valid
shrink$expectation
```

olive

*Fatty acid composition of Italian olive oils***Description**

Data on the percentage composition of eight fatty acids found by lipid fraction of 572 Italian olive oils. The data come from three areas; within each area there are a number of constituent regions, of which there are 9 in total.

**Usage**

```
data(olive)
```

**Format**

A data frame with 572 observations and 10 columns. The first columns gives the area (one of Southern Italy, Sardinia, and Northern Italy), the second gives the region, and the remaining 8 columns give the variables. Southern Italy comprises the North Apulia, Calabria, South Apulia, and Sicily regions, Sardinia is divided into Inland Sardinia and Coastal Sardinia and Northern Italy comprises the Umbria, East Liguria, and West Liguria regions.

**References**

Forina, M., Armanino, C., Lanteri, S. and Tiscornia, E. (1983). Classification of olive oils from their fatty acid composition, *Food Research and Data Analysis*, Applied Science Publishers, London, pp. 189-214.

PGMM\_dfree

*Estimate the Number of Free Parameters in Finite Factor Analytic Mixture Models (PGMM)*

**Description**

Estimates the dimension of the 'free' parameters in fully finite factor analytic mixture models, otherwise known as Parsimonious Gaussian Mixture Models (PGMM). This is used to calculate the penalty terms for the `aic.mcmc` and `bic.mcmc` model selection criteria implemented in [get\\_IMIFA\\_results](#) for *finite* factor models (though `mcmc_IMIFA` currently only implements UUU and UUC covariance structures).

**Usage**

```
PGMM_dfree(Q, P, G = 1, method = c("UUU", "UUC", "UCU", "UCC", "CUU", "CUC",
  "CCU", "CCC"))
```

**Arguments**

Q	The number of latent factors (which can be 0, corresponding to a model with diagonal covariance). This argument is vectorised.
P	The number of variables.
G	The number of groups. This defaults to 1.
method	By default, calculation assumes the UUU model with unconstrained loadings and unconstrained isotropic uniquenesses. The other seven models detailed in McNicholas and Murphy (2008) are also given. The first letter denotes whether loadings are constrained/unconstrained across groups; the second letter denotes the same for the uniquenesses; the final letter denotes whether uniquenesses are in turn constrained to be isotropic.

**Value**

A vector of length `length(Q)`.

**Note**

Though the function is available for standalone use, note that no checks take place, in order to speed up repeated calls to the function inside [mcmc\\_IMIFA](#).

**Author(s)**

Keefe Murphy

**References**

McNicholas, P. D. and Murphy, T. B. (2008) Parsimonious Gaussian Mixture Models, *Statistics and Computing*, 18(3): 285-296.

**See Also**

[get\\_IMIFA\\_results](#), [mcmc\\_IMIFA](#)

**Examples**

```
UUU <- PGMM_dfree(Q=4:5, P=50, G=3, method="UUU")
CCC <- PGMM_dfree(Q=4:5, P=50, G=3, method="CCC")
```

---

plot.Results_IMIFA	<i>Plotting output and parameters of inferential interest for IMIFA and related models</i>
--------------------	--

---

## Description

Plotting output and parameters of inferential interest for IMIFA and related models

## Usage

```
## S3 method for class 'Results_IMIFA'
plot(x = NULL, plot.meth = c("all", "correlation",
  "density", "errors", "GQ", "means", "parallel.coords", "trace", "zlabels"),
  param = c("means", "scores", "loadings", "uniquenesses", "pis", "alpha",
  "discount"), zlabels = NULL, heat.map = TRUE, palette = NULL,
  g = NULL, mat = TRUE, ind = NULL, fac = NULL, by.fac = FALSE,
  type = c("h", "n", "p", "l"), intervals = TRUE, partial = FALSE,
  titles = TRUE, transparency = 0.75, ...)
```

## Arguments

x	An object of class "Results_IMIFA" generated by <a href="#">get_IMIFA_results</a> .
plot.meth	The type of plot to be produced for the param of interest, where correlation refers to ACF/PACF plots, means refers to posterior means, density, trace and parallel.coords are self-explanatory. "all" in this case, the default, refers to trace, density, means, correlation. parallel.coords is only available when param is one of means, loadings or uniquenesses - note that this method applies a small amount of horizontal jitter to avoid overplotting. Special types of plots which don't require a param are GQ, for plotting the posterior summaries of the numbers of groups/factors, if available, zlabels for plotting clustering uncertainties if clustering has taken place (and, if available, the average similarity matrix, reorder according to the map labels) with or without the clustering labels being supplied via the zlabels argument), and errors for visualizing the difference between the estimated and empirical covariance matrix/matrices.
param	The parameter of interest for any of the following plot.meth options: trace, density, means, correlation. The param must have been stored when <a href="#">mcmc_IMIFA</a> was initially ran. Includes pis for methods where clustering takes place, and allows posterior inference on alpha and discount for the "IMFA" and "IMIFA" methods.
zlabels	The true labels can be supplied if they are known. If this is not supplied, the function uses the labels that were supplied, if any, to <a href="#">get_IMIFA_results</a> . Only relevant when plot.meth = "zlabels".
heat.map	Switch which allows plotting posterior mean loadings or posterior mean scores as a heatmap, or else as something akin to <code>link{plot(..., type="h")}</code> . Only relevant if param = "loadings" (in which case the default is TRUE) or param = "scores" (in which case the default is FALSE). Heatmaps are produced with the aid of <a href="#">mat2cols</a> and <a href="#">plot_cols</a> .

palette	An optional colour palette to be supplied if overwriting the default palette set inside the function by <a href="#">viridis</a> is desired.
g	Optional argument that allows specification of exactly which cluster the plot of interest is to be produced for. If not supplied, the user will be prompted to cycle through plots for all clusters. Also functions as an index for which plot to return when <code>plot.meth</code> is <code>GQ</code> or <code>zlabels</code> in much the same way.
mat	Logical indicating whether a <a href="#">matplot</a> is produced (defaults to <code>TRUE</code> ). If given as <code>FALSE</code> , <code>ind</code> is invoked.
ind	Either a single number indicating which variable to plot when <code>param</code> is one of means or uniquenesses, or which cluster to plot if <code>param</code> is <code>pis</code> . If scores are plotted, a vector of length two giving which observation and factor to plot; If loadings are plotted, a vector of length two giving which variable and factor to plot. Only relevant when <code>mat</code> or <code>by.fac</code> is <code>FALSE</code> .
fac	Optional argument that provides an alternative way to specify <code>ind[2]</code> when <code>mat</code> is <code>FALSE</code> and <code>param</code> is one of scores or loadings.
by.fac	Optionally allows (mat)plotting of scores and loadings by factor - i.e. observation(s) (scores) or variable(s) (loadings) for a given factor, respectively, controlled by <code>ind</code> or <code>fac</code> when set to <code>TRUE</code> . Otherwise factor(s) are plotted for a given observation or variable when set to <code>FALSE</code> (the default), again controlled by <code>ind</code> or <code>fac</code> . Only relevant when <code>param</code> is one of scores or loadings.
type	The manner in which the plot is to be drawn, as per the <code>type</code> argument to <a href="#">plot</a> .
intervals	Logical indicating whether credible intervals around the posterior mean(s) are to be plotted when <code>is.element(plot.meth, c("all", "means"))</code> . Defaults to <code>TRUE</code> .
partial	Logical indicating whether plots of type "correlation" use the PACF. The default, <code>FALSE</code> , ensures the ACF is used. Only relevant when <code>plot.meth = "all"</code> , otherwise both plots are produced when <code>plot.meth = "correlation"</code> .
titles	Logical indicating whether default plot titles are to be used ( <code>TRUE</code> ), or suppressed ( <code>FALSE</code> ).
transparency	A factor in <code>[0, 1]</code> modifying the opacity for overplotted lines. Defaults to <code>0.75</code> .
...	Other arguments typically passed to <a href="#">plot</a> .

**Value**

The desired plot with appropriate output and summary statistics printed to the console screen.

**Author(s)**

Keefe Murphy

**References**

Murphy, K., Gormley, I. C. and Viroli, C. (2017) Infinite Mixtures of Infinite Factor Analysers: Nonparametric Model-Based Clustering via Latent Gaussian Models, [arXiv:1701.07010](#).

**See Also**

[mcmc\\_IMIFA](#), [get\\_IMIFA\\_results](#), [mat2cols](#), [plot\\_cols](#)

**Examples**

```
# See the vignette associated with the package for more graphical examples:
# vignette("IMIFA", package = "IMIFA")

# data(olive)
# area      <- olive$area
# simIMIFA <- mcmc_IMIFA(olive, method="IMIFA")
# resIMIFA <- get_IMIFA_results(simIMIFA, z.avgsim=TRUE)

# Examine the posterior distribution(s) of the number(s) of clusters (G) &/or latent factors (Q)
# For the IM(I)FA and OM(I)FA methods, this also plots the trace of the active/non-empty clusters
# plot(resIMIFA, plot.meth="GQ")
# plot(resIMIFA, plot.meth="GQ", g=2)

# Plot clustering uncertainty (and, if available, the similarity matrix)
# plot(resIMIFA, plot.meth="zlabels", zlabels=area)

# Visualise empirical vs. estimated covariance error metrics
# plot(resIMIFA, plot.meth="errors")

# Look at the trace, density, posterior mean and correlation of various parameters of interest
# plot(resIMIFA, plot.meth="all", param="means", g=1)
# plot(resIMIFA, plot.meth="all", param="means", g=1, ind=2)
# plot(resIMIFA, plot.meth="all", param="scores")
# plot(resIMIFA, plot.meth="all", param="scores", by.fac=TRUE)
# plot(resIMIFA, plot.meth="all", param="loadings", g=1)
# plot(resIMIFA, plot.meth="all", param="loadings", g=1, heat.map=FALSE)
# plot(resIMIFA, plot.meth="parallel.coords", param="uniquenesses")
# plot(resIMIFA, plot.meth="all", param="pis", intervals=FALSE, partial=TRUE)
# plot(resIMIFA, plot.meth="all", param="alpha")
```

---

plot\_cols

*Plots a matrix of colours*

---

**Description**

Plots a matrix of colours as a heat map type image or as points. Intended for joint use with `mat2cols`.

**Usage**

```
plot_cols(cmat, na.col = "#808080FF", ptype = c("image", "points"),
  border.col = "#808080FF", dlabels = NULL, rlabels = FALSE,
  clabels = FALSE, pch = 15, cex = 3, label.cex = 0.6, ...)
```

**Arguments**

cmat	A matrix of valid colours, with missing values coded as NA allowed. Vectors should be supplied as matrices with 1 row or column, as appropriate.
na.col	Colour used for missing NA entries in cmat.
ptype	Switch controlling output as either a heat map "image" (the default) or as "points".
border.col	Colour of border drawn around the plot.
dlabels	Vector of labels for the diagonals.
rlabels	Vector of labels for the rows.
clabels	Vector of labels for the columns.
pch	Point type used when ptype="points".
cex	Point cex used when ptype="points".
label.cex	Govens cex parameter used for labels.
...	Further graphical parameters.

**Value**

Either an "image" or "points" plot of the supplied colours.

**See Also**

[mat2cols](#), [image](#), [heat\\_legend](#), [is.cols](#)

**Examples**

```
# Generate a colour matrix using mat2cols()
mat      <- matrix(rnorm(100), nrow=10, ncol=10)
mat[2,3] <- NA
cols     <- heat.colors(12)[12:1]
matcol   <- mat2cols(mat, cols=cols)
matcol

# Use plot_cols() to visualise the colours matrix
par(mar=c(5.1, 4.1, 4.1, 4.1))
plot_cols(matcol)

# Add a legend using heat_legend()
heat_legend(mat, cols=cols); box(lwd=2)
```

---

Procrustes

*Procrustes Transformation*


---

### Description

This function performs a Procrustes transformation on a matrix  $X$  to minimize the squared distance between  $X$  and another comparable matrix  $X_{\text{star}}$ .

### Usage

```
Procrustes(X, Xstar, translate = FALSE, dilate = FALSE, sumsq = FALSE)
```

### Arguments

$X$	The matrix to be transformed.
$X_{\text{star}}$	The target matrix.
<code>translate</code>	Logical value indicating whether $X$ should be translated (defaults to FALSE).
<code>dilate</code>	Logical value indicating whether $X$ should be dilated (defaults to FALSE).
<code>sumsq</code>	Logical value indicating whether the sum of squared differences between $X$ and $X_{\text{star}}$ should be calculated and returned.

### Details

$R$ ,  $t$ , and  $d$  are chosen so that:

$$d \times \mathbf{X}\mathbf{R} + \mathbf{1}t^{\top} \approx X^{\star}$$

$X_{\text{new}}$  is given by:

$$X_{\text{new}} = d \times \mathbf{X}\mathbf{R} + \mathbf{1}t^{\top}$$

### Value

A list containing:

**$X_{\text{new}}$**  The matrix that is the Procrustes transformed version of  $X$ .

**$R$**  The rotation matrix.

**$t$**  The translation vector (if `isTRUE(translate)`).

**$d$**  The scaling factor (if `isTRUE(dilate)`).

**$ss$**  The sum of squared differences (if `isTRUE(sumsq)`).

### References

Borg, I. and Groenen, P. J. F. (1997) *Modern Multidimensional Scaling*. Springer-Verlag, New York, pp. 340-342.



**Examples**

```
# Match two matrices, allowing translation and dilation
mat1 <- diag(rnorm(10))
mat2 <- 0.05 * matrix(rnorm(100), 10, 10) + mat1
proc <- Procrustes(X=mat1, Xstar=mat2, translate=TRUE, dilate=TRUE, sumsq=TRUE)

# Extract the transformed matrix, rotation matrix, translation vector and scaling factor
mat_new <- proc$X.new
mat_rot <- proc$R
mat_t <- proc$t
mat_d <- proc$d

# Compare the sum of squared differences to a Procrustean transformation with rotation only
mat_ss <- proc$ss
mat_ss2 <- Procrustes(X=mat1, Xstar=mat2, sumsq=TRUE)$ss
```

---

psi_hyper	<i>Find sensible inverse gamma hyperparameters for variance/uniqueness parameters</i>
-----------	---

---

**Description**

Takes a shape hyperparameter and covariance matrix, and finds data-driven rate hyperparameters in such a way that Heywood problems are avoided for factor analysis or probabilistic principal components analysis (and mixtures thereof). Rates are allowed to be variable-specific or a single value under the factor analysis model, but must be a single value for the PPCA model. Used internally by `mcmc_IMIFA` when its argument `psi_beta` is not supplied.

**Usage**

```
psi_hyper(shape, covar, type = c("unconstrained", "isotropic"))
```

**Arguments**

shape	A positive shape hyperparameter.
covar	A square, positive-semidefinite covariance matrix.
type	A switch indicating whether a single rate ( <code>isotropic</code> ) or variable-specific rates ( <code>unconstrained</code> ) are to be derived. The isotropic constraint provides the link between factor analysis and the probabilistic principal components analysis model. Uniquenesses are only allowed to be variable specific under the factor analysis model.

**Value**

Either a single rate hyperparameter or `ncol(covar)` variable specific hyperparameters.

**Author(s)**

Keefe Murphy

**References**

Fruwirth-Schnatter, S. and Lopes, H. F. (2010). Parsimonious Bayesian factor analysis when the number of factors is unknown, *Technical Report*. The University of Chicago Booth School of Business.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3): 611-622.

**See Also**

[mcmc\\_IMIFA](#)

**Examples**

```
data(olive)
olive2 <- olive[,-(1:2)]
rates <- psi_hyper(shape=2.5, covar=cov(olive2), type="isotropic")
rates

olive_scaled <- scale(olive2, center=TRUE, scale=TRUE)
rate <- psi_hyper(shape=3, covar=cov(olive_scaled), type="unconstrained")
rate
```

---

rDirichlet

---

*Simulate Mixing Proportions from a Dirichlet Distribution*


---

**Description**

Generates samples from the Dirichlet distrubution with parameter alpha efficiently by simulating Gamma(alpha, 1) random variables and normalising them.

**Usage**

```
rDirichlet(G, alpha, nn = 0)
```

**Arguments**

G	The number of groups for which weights need to be sampled.
alpha	The Dirichlet hyperparameter, either of length 1 or G. When the length of alpha is 1, this amounts to assuming an exchangeable prior. Be warned that this will be recycled if necessary.
nn	A vector giving the number of observations in each of G groups so that Dirichlet posteriors rather than priors can be sampled from. This defaults to 0, i.e. simulation from the prior. Be warned that this will be recycled if necessary.

**Value**

A Dirichlet vector of G weights which sum to 1.

**Note**

Though the function is available for standalone use, note that no checks take place, in order to speed up repeated calls to the function inside `mcmc_IMIFA`.

**References**

Devroye, L. (1986) *Non-Uniform Random Variate Generation*, Springer-Verlag, New York, p. 594.

**Examples**

```
prior      <- rDirichlet(G=5, alpha=1)
posterior <- rDirichlet(G=5, alpha=1, nn=c(20, 41, 32, 8, 12))
```

---

shift\_GA

---

*Moment Matching Parameters of Shifted Gamma Distributions*


---

**Description**

This function takes shape and rate parameters of a Gamma distribution and modifies them to achieve the same expected value and variance when the left extent of the support of the distribution is shifted up or down.

**Usage**

```
shift_GA(shape, rate, shift = 0L, param = c("rate", "scale"))
```

**Arguments**

shape	Shape parameter a of a Gamma(a, b) distribution. Must be strictly positive.
rate	Rate parameter b of a Gamma(a, b) distribution. Must be strictly positive.
shift	Modifier, such that the Gamma distribution has support on (shift, $\infty$ ). Can be positive or negative, though typically negative and small.
param	Switch controlling whether the supplied rate parameter is indeed a rate, or actually a scale parameter. Also governs whether the output is given in terms of rate or scale. Defaults to "rate".

**Value**

A list of length 2, containing the modified shape and rate parameters, respectively.

**Author(s)**

Keefe Murphy

### Examples

```
# Shift a Ga(shape=4, rate=2) distribution to the left by 1;
# achieving the same expected value of 2 and variance of 1.
shift_GA(4, 2, -1)
```

---

sim\_IMIFA\_data

---

*Simulating Data from a Mixture of Factor Analysers Structure*


---

### Description

Function to simulate data of any size and dimension from a mixture of (infinite) factor analysers structure.

### Usage

```
sim_IMIFA_data(N = 300L, G = 3L, P = 50L, Q = rep(floor(log(P)), G),
  pis = rep(1/G, G), psi = NULL, nn = NULL, loc.diff = 1L,
  method = c("conditional", "marginal"))
```

### Arguments

N	Desired overall number of observations in the simulated data set - a single integer.
G	Desired number of clusters in the simulated data set - a single integer.
P	Desired number of variables in the simulated dataset - a single integer.
Q	Desired number of cluster-specific latent factors in the simulated data set. Can be specified either as a single integer if all clusters are to have the same number of factors, or a vector of length G. Defaults to <code>floor(log(P))</code> in each group.
pis	Mixing proportions of the clusters in the dataset if $G > 1$ . Must sum to 1. Defaults to <code>rep(1/G, G)</code> .
psi	True values of uniqueness parameters, either as a single value, a vector of length G, a vector of length P, or a $G \times P$ matrix: as such the user can specify uniquenesses as a diagonal or isotropic matrix, and further constrain uniquenesses across groups if desired. If <code>psi</code> is missing, uniquenesses are simulated via <code>rgamma(P, 1, 1)</code> within each group.
nn	An alternative way to specify the size of each cluster, by giving the exact number of observations in each group explicitly. Must sum to N.
loc.diff	A parameter to control the closeness of the clusters in terms of the difference in their location vectors. Defaults to 1.
method	A switch indicating whether the mixture to be simulated from is the conditional distribution of the data given the latent variables (default), or simply the marginal distribution of the data.

**Value**

Invisibly returns a data.frame with N observations (rows) of P variables (columns). The true values of the parameters which generated these data are also stored.

**Author(s)**

Keefe Murphy

**See Also**

The function `mcmc_IMIFA` for fitting an IMIFA related model to the simulated data set.

**Examples**

```
# Simulate 100 observations from 3 balanced groups with cluster-specific numbers of latent factors
# Specify isotropic uniquenesses within each cluster
sim_data <- sim_IMIFA_data(N=100, G=3, P=20, Q=c(2, 2, 5), psi=1:3)
names(attributes(sim_data))
labels <- attr(sim_data, "Labels")

# Visualise the data in two-dimensions
plot(cmdscale(dist(sim_data), k=2), col=labels)

# Fit a MIFA model to this data
# tmp <- mcmc_IMIFA(sim_data, method="MIFA", range.G=3, n.iters=5000)
```

---

Zsimilarity

*Summarises MCMC clustering labels with a similarity matrix and finds the 'average' clustering*

---

**Description**

This functions takes a Monte Carlo sample of cluster labels, converts them to adjacency matrices, and computes a similarity matrix as an average of the adjacency matrices. The dimension of the similarity matrix is invariant to label switching and the number of clusters in each sample. As a summary of the posterior clustering, the index of the clustering with minimum squared distance to this 'average' clustering is reported.

**Usage**

```
Zsimilarity(zs)
```

**Arguments**

**zs** A matrix containing samples of clustering labels where the rows correspond to the number of observations and the columns correspond to the number of iterations.

**Value**

A list containing three elements:

**z.avg** The 'average' clustering, with minimum squared distance to `z.sim`.

**z.sim** The  $N \times N$  similarity matrix, in a sparse format (see [as.simple\\_triplet\\_matrix](#)). If the data have been previously ordered, a (ordered) heatmap may provide a useful visualisation. The user is also invited to perform hierarchical clustering using [hclust](#) after first converting this similarity matrix to a distance matrix - "complete" linkage is recommended.

**dist.z** A vector of length  $N$  recording the distances between each clustering and the 'average' clustering.

**Note**

This function is implemented purely in R and as such its performance in terms of speed and memory may not be optimal; it can take quite a considerable amount of time to run, and may crash if the number of observations &/or number of iterations is so large that the similarity matrix is insufficiently sparse. This function can optionally be called inside [get\\_IMIFA\\_results](#).

**Author(s)**

Keefe Murphy

**See Also**

[get\\_IMIFA\\_results](#), [as.simple\\_triplet\\_matrix](#), [hclust](#)

**Examples**

```
# Run a IMIFA model and extract the sampled cluster labels
# data(olive)
# sim    <- mcmc_IMIFA(olive, method="IMIFA", n.iters=5000)
# zs     <- sim[[1]][[1]]$z.store

# Get the similarity matrix and visualise it
# zsimil <- Zsimilarity(zs)
# z.sim  <- as.matrix(zsimil$z.sim)
# z.sim2 <- replace(z.sim, z.sim == 0, NA)
# image(z.sim2, col=heat.colors(30)[30:1]); box(lwd=2)

# Extract the clustering with minimum squared distance to this
# 'average' and evaluate its performance against the true labels
# z.avg  <- zsimil$z.avg
# table(z.avg, olive$area)

# Perform hierarchical clustering on the distance matrix
# Hcl    <- hclust(as.dist(1 - z.sim), method="complete")
# plot(Hcl)
# hier.z <- cutree(Hcl, k=3)
# table(hier.z, olive$area)
```

# Index

## \*Topic **datasets**

- coffee, [2](#)
- olive, [26](#)

as.simple\_triplet\_matrix, [38](#)

coffee, [2](#)

G\_expected, [7](#), [8–10](#)

G\_priorDensity, [8](#), [8](#), [10](#)

G\_variance, [8](#), [9](#), [10](#)

get\_IMIFA\_results, [3](#), [12](#), [16](#), [18](#), [23](#), [26–28](#),  
[30](#), [38](#)

gumbel\_max, [6](#)

hclust, [38](#)

heat\_legend, [11](#), [15](#), [31](#)

image, [11](#), [31](#)

IMIFA, [11](#)

IMIFA-package (IMIFA), [11](#)

is.cols, [11](#), [13](#), [15](#), [31](#)

is.posi\_def, [13](#)

Ledermann, [14](#)

loadings, [5](#)

mat2cols, [11](#), [15](#), [28](#), [30](#), [31](#)

matplot, [29](#)

Mclust, [19](#)

mcmc\_IMIFA, [3–8](#), [10](#), [12](#), [16](#), [24–28](#), [30](#),  
[33–35](#), [37](#)

MGP\_check, [20](#), [23](#), [24](#)

olive, [26](#)

PGMM\_dfree, [19](#), [26](#)

plot, [29](#)

plot.Results\_IMIFA, [4](#), [5](#), [12](#), [15](#), [28](#)

plot\_cols, [11](#), [15](#), [28](#), [30](#), [30](#)

Procrustes, [5](#), [32](#)

psi\_hyper, [19](#), [23](#), [33](#)

rDirichlet, [34](#)

Rmpfr, [8–10](#)

rowLogSumExps, [6](#), [7](#)

shift\_GA, [35](#)

sim\_IMIFA\_data, [36](#)

viridis, [15](#), [29](#)

Zsimilarity, [4](#), [5](#), [37](#)