Package 'IMIFA'

January 5, 2021

Type Package

Date 2020-12-29

Title Infinite Mixtures of Infinite Factor Analysers and Related Models

Version 2.1.5

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. (2020) <doi:10.1214/19-BA1179>. 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, conducting posterior inference on parameters of interest, posterior predictive checking, and quantifying uncertainty.

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

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

BugReports https://github.com/Keefe-Murphy/IMIFA

LazyData true

Imports matrixStats (>= 0.53.1), mclust (>= 5.4), mvnfast, Rfast (>= 1.9.8), slam, viridisLite

Suggests gmp (>= 0.5-4), knitr, mcclust, rmarkdown, Rmpfr

RoxygenNote 7.1.1

VignetteBuilder knitr

Collate 'MainFunction.R' 'Diagnostics.R' 'FullConditionals.R' 'Gibbs_FA.R' 'Gibbs_IFA.R' 'Gibbs_IMFA.R' 'Gibbs_IMIFA.R' 'Gibbs_MFA.R' 'Gibbs_MIFA.R' 'Gibbs_OMFA.R' 'Gibbs_OMIFA.R' 'IMIFA.R' 'PlottingFunctions.R' 'SimulateData.R' 'data.R'

NeedsCompilation no

Author Keefe Murphy [aut, cre] (<https://orcid.org/0000-0002-7709-3159>),
Cinzia Viroli [ctb] (<https://orcid.org/0000-0002-3278-5266>),
Isobel Claire Gormley [ctb] (<https://orcid.org/0000-0001-7713-681X>)

Maintainer Keefe Murphy <keefe.murphy@mu.ie>

Repository CRAN

Date/Publication 2021-01-04 23:10:02 UTC

R topics documented:

IMIFA-package *IMIFA: Infinite Mixtures of Infinite Factor Analysers and Related Models*

Description

A package for Bayesian nonparameteric clustering of high-dimensional data sets, providing functions for fitting, diagnostic tools and plotting for Infinite Mixtures of Infinite Factor Analysers and the full suite of related models introduced by Murphy et al. (2020) <doi: 10.1214/19BA1179>. 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, conducting posterior inference on parameters of interest, posterior predictive checking, and quantifying uncertainty.

Details

- Type: Package
- Package: IMIFA
- Version: 2.1.5
- Date: 2020-12-29 (this version), 2017-02-02 (original release)
- Licence: GPL (>=2)

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, e.g. for simulating data from a multivariate mixture of factor analysers, many functions for soliciting good priors, and many functions related to plotting.

`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., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

See Also

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

Author(s)

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

Maintainer: Keefe Murphy - <<keefe.murphy@mu.ie>>

See Also

Useful links:

- https://cran.r-project.org/package=IMIFA
- Report bugs at https://github.com/Keefe-Murphy/IMIFA

`bnpControl` *Control settings for the Bayesian Nonparametric priors for infinite mixture models (or shrinkage priors for overfitted mixtures)*

Description

Supplies a list of arguments for use in `mcmc_IMIFA` pertaining to the use of the Bayesian Nonparametric Pitman-Yor / Dirichlet process priors with the infinite mixture models `"IMFA"` and `"IMIFA"`. Certain arguments related to the Dirichlet concentration parameter for the overfitted mixtures `"OMFA"` and `"OMIFA"` can be supplied in this manner also.

Usage

```
bnpControl(learn.alpha = TRUE,
           alpha.hyper = c(2L, 4L),
           discount = 0,
           learn.d = TRUE,
           d.hyper = c(1L, 1L),
           ind.slice = TRUE,
           rho = 0.75,
           trunc.G = NULL,
           kappa = 0.5,
           IM.lab.sw = TRUE,
           zeta = NULL,
           tune.zeta = list(...),
           ...)
```

Arguments

`learn.alpha`

For the `"IMFA"` **and** `"IMIFA"` **methods:** A logical indicating whether the Pitman-Yor / Dirichlet process concentration parameter is to be learned (defaults to `TRUE`), 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 when `discount > 0`. If not being learned, `alpha` *must* be supplied.
In the special case of `discount < 0`, `alpha` must be supplied as a positive integer multiple of `abs(discount)`; in this instance, `learn.alpha` is forced to `TRUE` and `alpha` is updated with the changing number of components as the positive integer.

For the `"OMFA"` **and** `"OMIFA"` **methods:** A logical indicating whether the Dirichlet concentration parameter is to be learned (defaults to `TRUE`) or remain fixed for the duration of the chain. If being learned, a Ga(a, b * G) is assumed for `alpha`, where G is the number of mixture components `range.G`, and updates take place via Metropolis-Hastings. If not being learned `alpha` *must* be supplied.

`alpha.hyper`

For the `"IMFA"` **and** `"IMIFA"` **methods:** A vector of length 2 giving hyperparameters for the prior on the Pitman-Yor / Dirichlet process concentration parameter `alpha`. If `isTRUE(learn.alpha)`, these are shape and rate parameters of a Gamma distribution. Defaults to Ga(`2`, `4`). Choosing a larger rate is particularly important, as it encourages clustering. The prior is shifted to have support on (`-discount`, `Inf`) when non-zero `discount` is supplied and remains fixed (i.e. `learn.d=FALSE`) or when `learn.d=TRUE`.

For the `"OMFA"` **and** `"OMIFA"` **methods:** A vector of length 2 giving hyperparameters a and b for the prior on the Dirichlet concentration parameter `alpha`. If `isTRUE(learn.alpha)`, these are shape and rate parameters of a Gamma distribution. Defaults to Ga(2, 4). Note that the supplied rate will be multiplied by `range.G`, to encourage clustering, such that the form of the prior is Ga(a, b * G).

`discount`

The discount parameter used when generalising the Dirichlet process to the Pitman-Yor process. Defaults to 0, but typically must lie in the interval [0, 1). If greater than zero, `alpha` can be supplied greater than `-discount`. By default, Metropolis-Hastings steps are invoked for updating this parameter via `learn.d`.

The special case of `discount < 0` is allowed, in which case `learn.d=FALSE` is forced and `alpha` must be supplied as a positive integer multiple of `abs(discount)`. Fixing `discount > 0.5` is discouraged (see `learn.alpha`).

`learn.d`

Logical indicating whether the `discount` parameter is to be updated via Metropolis-Hastings (defaults to `TRUE`, unless `discount` is supplied as a negative value).

`d.hyper`

Hyperparameters for the Beta(a,b) prior on the `discount` parameter. Defaults to Beta(1,1), i.e. Uniform(0,1).

`ind.slice`

Logical indicating whether the independent slice-efficient sampler is to be employed (defaults to `TRUE`). If `FALSE` the dependent slice-efficient sampler is employed, whereby the slice sequence $\xi_1, \ldots, \xi_g$ is equal to the decreasingly ordered mixing proportions.

rho — Parameter controlling the rate of geometric decay for the independent slice-efficient sampler, s.t. $\xi = (1-\rho)\rho^{g-1}$. 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 $\xi_1, \ldots, \xi_g$ is equal to the *expectation* of the decreasingly ordered mixing proportions. Only relevant when `ind.slice` is `TRUE`.

trunc.G — The maximum number of allowable and storable clusters under the `"IMIFA"` and `"IMFA"` models. The number of active clusters to be sampled at each iteration is adaptively truncated, with `trunc.G` as an upper limit for storage reasons. Defaults to `max(min(N-1,50),range.G))` and must satisfy `range.G <= trunc.G < N`. Note that large values of `trunc.G` may lead to memory capacity issues.

kappa — 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. `kappa` gives the weight of the point mass at zero (the 'spike'). Must lie in the interval [0,1]. Defaults to 0.5. Only relevant when `isTRUE(learn.d)`. A value of 0 ensures non-zero discount values (i.e. Pitman-Yor) at all times, and *vice versa*. Note that `kappa` will default to exactly 0 if `alpha<=0` and `learn.alpha=FALSE`.

IM.lab.sw — Logical indicating whether the two forced label switching moves are to be implemented (defaults to `TRUE`) when running one of the infinite mixture models.

zeta — **For the `"IMFA"` and `"IMIFA"` methods:** Tuning parameter controlling the acceptance rate of the random-walk proposal for the Metropolis-Hastings steps when `learn.alpha=TRUE`, where `2 * zeta` gives the full width of the uniform proposal distribution. These steps are only invoked when either `discount` is non-zero and fixed or `learn.d=TRUE`, otherwise `alpha` is learned by Gibbs updates. Must be strictly positive (if invoked). Defaults to `2`.

For the `"OMFA"` and `"OMIFA"` methods: Tuning parameter controlling the standard deviation of the log-normal proposal for the Metropolis-Hastings steps when `learn.alpha=TRUE`. Must be strictly positive (if invoked). Defaults to `0.75`.

tune.zeta — A list with the following named arguments, used for tuning `zeta` (which is either the width of the uniform proposal for the `"IMFA"` or `"IMIFA"` methods or the standard deviation of the log-normal proposal for the `"OMFA"` or `"OMIFA"` methods) for `alpha`, via diminishing Robbins-Monro type adaptation, when the `alpha` parameter is learned via Metropolis-Hastings steps:

`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 `tune.zeta` are.

`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 `tune.zeta` are.

`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.

`start.zeta` The iteration at which diminishing adaptation begins. Defaults to `100`.

`stop.zeta` The iteration at which diminishing adaptation is to stop completely. Defaults to `Inf`, such that diminishing adaptation is never explicitly made to stop. Must be greater than `start.zeta`.

At least one `tune.zeta` argument must be supplied for diminishing adaptation to be invoked. `tune.zeta` arguments are only relevant when `learn.alpha` is `TRUE` (and, for the `"IMFA"` and `"IMIFA"` methods, when either of the following is also true: the `discount` remains fixed at a non-zero value, or when `learn.d` is `TRUE` and `kappa < 1`). Since Gibbs steps are invoked for updating `alpha` when `discount == 0` under the `"IMFA"` or `"IMIFA"` methods, adaption occurs according to a running count of the number of iterations with non-zero sampled `discount` values for those methods.

If diminishing adaptation is 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.

`...` Catches unused arguments.

Details

The crucial concentration parameter `alpha` is documented within the main `mcmc_IMIFA` function, and is relevant to all of the `"IMIFA"`, `"IMFA"`, `"OMIFA"`, and `"OMFA"` methods.

All arguments here are relevant to the `"IMFA"` and `"IMIFA"` methods, but the following are also related to the `"OMFA"` and `"OMIFA"` methods, and may behave differently in those instances: `learn.alpha`, `alpha.hyper`, `zeta`, and `tune.zeta`.

Value

A named list in which the names are the names of the arguments related to the BNP prior(s) and the values are the values supplied to the arguments.

Note

Certain supplied arguments will be subject to further checks within `mcmc_IMIFA`. `G_priorDensity` and `G_moments` can help with soliciting sensible DP/PYP priors.

Under the `"IMFA"` and `"IMIFA"` methods, a Pitman-Yor process prior is specified by default. A Dirichlet process prior can be easily invoked when the `discount` is fixed at `0` and `learn.d=FALSE`. The normalized stable process can also be specified as a prior distribution, as a special case of the Pitman-Yor process, when `alpha` remains fixed at `0` and `learn.alpha=FALSE` (provided the `discount` is fixed at a strictly positive value or `learn.d=TRUE`). The special case of the Pitman-Yor process with negative `discount` is also allowed as an experimental feature for which caution is advised, though `learn.d` and `learn.alpha` are forced to `FALSE` and `TRUE`, respectively, in this instance.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

Kalli, M., Griffin, J. E. and Walker, S. G. (2011) Slice sampling mixture models, *Statistics and Computing*, 21(1): 93-105.

See Also

`mcmc_IMIFA`, `G_priorDensity`, `G_moments`, `mixfaControl`, `mgpControl`, `storeControl`

Examples

```
bnpctrl <- bnpControl(learn.d=FALSE, ind.slice=FALSE, alpha.hyper=c(3, 3))

# data(olive)
# sim   <- mcmc_IMIFA(olive, "IMIFA", n.iters=5000, BNP=bnpctrl)

# Alternatively specify these arguments directly
# sim   <- mcmc_IMIFA(olive, "IMIFA", n.iters=5000, learn.d=FALSE,
#                     ind.slice=FALSE, alpha.hyper=c(3, 3))
```

---

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 Kaffee-Chemie, *Association Scientifique International du Cafe, 6th International Colloquium on Coffee Chemistry*, Bogata, Columbia, pp. 61-72.

Examples

```
data(coffee, package="IMIFA")
pairs(coffee[,-(1:2)], col=coffee$Variety)
```

`get_IMIFA_results` *Extract results, conduct posterior inference and compute performance metrics for MCMC samples of models from the IMIFA family*

Description

This function post-processes simulations generated by `mcmc_IMIFA` for any of the IMIFA family of models. This includes accounting for label switching, and accounting for rotational invariance via Procrustean methods. 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. Posterior predictive checking of the appropriateness of the fitted model is also facilitated.

Usage

```
get_IMIFA_results(sims = NULL,
                  burnin = 0L,
                  thinning = 1L,
                  G = NULL,
                  Q = NULL,
                  criterion = c("bicm", "aicm", "dic", "bic.mcmc", "aic.mcmc"),
                  G.meth = c("mode", "median"),
                  Q.meth = c("mode", "median"),
                  conf.level = 0.95,
                  error.metrics = TRUE,
                  vari.rot = FALSE,
                  z.avgsim = FALSE,
                  zlabels = NULL,
                  nonempty = TRUE,
                  ...)

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

## S3 method for class 'Results_IMIFA'
summary(object,
        MAP = TRUE,
        ...)
```

Arguments

sims	An object of class "IMIFA" generated by mcmc_IMIFA.
burnin	Optional additional number of iterations to discard. Defaults to 0, corresponding to no additional burnin. See mixfaControl for the default burnin settings used previously by mcmc_IMIFA.
thinning	Optional interval for extra thinning to be applied. Defaults to 1, corresponding to no additional thinning. See mixfaControl for the default thinning settings used previously by mcmc_IMIFA.
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 not 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. Can be supplied as a scalar or a vector of values for each cluster.
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 sims was created via mcmc_IMIFA. Defaults to bicm, but note that these are *all* 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. The aic.mcmc and bic.mcmc criteria will only be returned for finite factor models.
G.meth	If the object in sims 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". Often the mode and median will agree in any case.
Q.meth	If the object in sims 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". Often the mode and median will agree in any case.
conf.level	The confidence level to be used throughout for credible intervals for all parameters of inferential interest, and error metrics if error.metrics=TRUE. Defaults to 0.95.
error.metrics	A logical activating or deactivating posterior predictive checking: i.e. controlling whether metrics quantifying a) the posterior predictive reconstruction error (PPRE) between bin counts of the data and bin counts of replicate draws from

the posterior distribution & and b) the error between the empirical and estimated covariance matrices should be computed. These are computed for every *valid* retained iteration (see `Details`). Defaults to `TRUE`, but can be time-consuming for models which achieve clustering. These error metrics, and the uncertainty associated with them, can be visualised via `plot.Results_IMIFA`. Depending on what parameters were stored when calling `mcmc_IMIFA`, potentially not all error metrics will be available to compute.

The Frobenius norm is used in the computation of the PPRE, by default, but the `type` of `norm` can be changed via the `...` construct below. So too can the breakpoints (`dbreaks`) used to bin the data and the posterior predictive replicate data sets. Some caution is advised in the latter case.

`vari.rot` Logical indicating whether the loadings matrix/matrices template(s) should be `varimax` rotated first, prior to the Procrustes rotation steps. Defaults to `FALSE`. Not necessary at all for clustering purposes, or inference on the covariance matrix, but useful if interpretable inferences on the loadings matrix/matrices are desired. Arguments to `varimax` can be passed via the `...` construct, but note that the argument `normalize` here defaults to `FALSE`.

`z.avgsim` Logical (defaults to `FALSE`) indicating whether the clustering should also be summarised with a call to `Zsimilarity` by the clustering with minimum mean squared error to the similarity matrix obtained by averaging the stored adjacency matrices, in addition to the MAP estimate.

Note that the MAP clustering is computed *conditional* on the estimate of the number of clusters (whether that be the modal estimate or the estimate according to `criterion`) 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 specifying `z.avgsim=TRUE` may not necessarily coincide with the MAP estimate of `G`, but it may provide a useful alternative summary of the partitions explored during the chain, and the user is free to call `get_IMIFA_results` again with the new suggested `G` value.

Please be warned that this feature requires loading the `mcclust` package. This is liable to 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. When `z.avgsim=TRUE`, both the summarised clustering and the similarity matrix are stored: the latter can be visualised as part of a call to `plot.Results_IMIFA`.

`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.

`nonempty` For `"MFA"` and `"MIFA"` models ONLY: a logical indicating whether only iterations with non-empty components should be retained. Defaults to `TRUE`, but may lead to empty chains - conversely, `FALSE` may lead to empty components and related errors.

`x, object, MAP, ...`

Arguments required for the `print.Results_IMIFA` and `summary.Results_IMIFA` functions: `x` and `object` are objects of class `"Results_IMIFA"` resulting from a call to `get_IMIFA_results`. `MAP` is a logical which governs whether a table

of the MAP classification is printed, while ... gathers additional arguments to those functions.

Users can also pass the type argument to the norm function when isTRUE(error.metrics) and the posterior predictive reconstruction error (PPRE) is calculated. By default the Frobenius norm (type="F") is employed.

Finally, the ... construct also allows arguments to varimax to be passed to get_IMIFA_results itself, when isTRUE(vari.rot), or arguments to hist when isTRUE(error.metrics), in order to guide construction of the bins. Additionally, by passing the argument dbreaks via the ... construct, the bins can be specified directly. However, caution is advised in doing so; in particular, the bins must be constructed on data which has been standardised in the same way as the data modelled within mcmc_IMIFA.

Details

The function also performs post-hoc corrections for label switching, as well as post-hoc Procrustes rotation of loadings matrices and scores, in order to ensure sensible posterior parameter estimates, computes error metrics, constructs credible intervals, and generally transforms the raw sims object into an object of class "Results_IMIFA" in order to prepare the results for plotting via plot.Results_IMIFA.

For the infinite factor methods, iterations where the maximum number of factors was greater than or equal to the maximum of the estimated cluster-specific factors are retained for posterior summaries of the scores, in order to preserve the estimated dimension of the scores matrices. Similarly, these are also the *valid* iterations used for the computation of the averages and credible intervals for the error metrics. For the finite factor models, *all* retained iterations are used in both instances (i.e. both for the scores and the error metrics).

In all cases, only iterations with G non-empty components are retained.

Value

An object of class "Results_IMIFA" to be passed to plot.Results_IMIFA for visualising results. Dedicated print and summary functions also 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 (or models where the estimate number of clusters is 1), in particular x$Clust$MAP, the MAP summary of the posterior clustering, the last valid sample of cluster labels x$Clust$last.z, the matrix of posterior cluster membership probabilities x$Clust$post.prob, and the posterior confusion matrix x$Clust$PCM.

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. Posterior summaries of the mixing proportions, and the concentration/discount parameters, if necessary, are also included here, as well as the last valid samples of each.

Error — Everything pertaining the model fit assessment can be found here, incl. the distribution of the PPRE values and associated bin counts for the replicate draws, as

well as average error metrics (e.g. MSE, RMSE), and credible intervals quantifying the associated uncertainty, between the empirical and estimated covariance matrix/matrices, both of which are also included.

`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. Model selection criterion values are also accessible here.

`Means` Posterior summaries for the means, after conditioning on `G`.

`Loadings` Posterior summaries for the factor loadings matrix/matrices, after conditioning on `G` and `Q`. 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.

The number of iterations retained for posterior summaries of the loadings may vary for different clusters for the infinite factor methods, corresponding to iterations where the cluster-specific number of factors was greater than or equal to the modal estimate of the cluster-specific number of factors.

`Scores` Posterior summaries for the latent factor scores, after conditioning on the maximum of the estimated number of cluster-specific factors. Summaries are given for the *single* matrix of factor scores. See `scores_MAP` to decompose these summaries into sub-matrices according to the MAP partition (for models which achieve clustering).

For the infinite factor methods, iterations where the maximum number of factors was greater than or equal to the maximum of the estimated cluster-specific factors are retained for posterior summaries of the scores, in order to preserve the estimated dimension of the scores matrices.

`Uniquenesses` Posterior summaries for the uniquenesses, after conditioning on `G`.

The objects `Means`, `Loadings`, `Scores` and `Uniquenesses` (if stored when calling `mcmc_IMIFA`!) also contain, as well as the posterior summaries, the entire chain of valid samples of each, as well as, for convenience, the last valid samples of each (after conditioning on the modal `G` and `Q` values, and accounting for label switching, and rotational invariance via Procrustes rotation).

Note

For the `"IMIFA"`, `"IMFA"`, `"OMIFA"`, and `"OMFA"` methods, the retained mixing proportions are renormalised after conditioning on the modal `G`. This is especially necessary for the computation of the `error.metrics`, just note that the values on which posterior inference are conducted will ever so slightly differ from the actually sampled values.

Due to the way the offline label-switching correction is performed, different runs of this function may give *very slightly* different results in terms of the cluster labellings (and by extension the parameters, which are permuted in the same way), but only if the chain was run for an extremely small number of iterations, well below the number required for convergence, and samples of the cluster labels match poorly across iterations (particularly if the number of clusters suggested by those sampled labels is high).

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

See Also

plot.Results_IMIFA, mcmc_IMIFA, Zsimilarity, scores_MAP, sim_IMIFA_model, Procrustes, varimax, norm

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=1000)

# 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)

# Simulate new data from the above model
# newdata       <- sim_IMIFA_model(resIMIFAolive)
```

---

gumbel_max *Simulate Cluster Labels from Unnormalised Log-Probabilities using the Gumbel-Max Trick*

Description

Samples cluster labels for N observations from G clusters 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.

Usage

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

Arguments

probs	An N x G matrix of unnormalised probabilities on the log scale, where N is he 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. Typically N > 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 mcmc_IMIFA. Details of this correction are given in Murphy et. al. (2020). When set to TRUE, this results in a speed-improvement when probs contains non-finite values (e.g. -Inf, corresponding to zero on the probability scale).

Details

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.

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 - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

Yellott, 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(2): 109-144.

See Also

mcmc_IMIFA, rowLogSumExps

Examples

```
# Create weights for 3 components
  G         <- 3
  weights   <- seq_len(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_moments *1st & 2nd Moments of the Pitman-Yor / Dirichlet Processes*

Description

Calculates the *a priori* expected number of clusters or the variance of the number of clusters under a PYP or DP prior for a sample of size `N` at given values of the concentration parameter `alpha` and optionally also the Pitman-Yor `discount` parameter. Useful for soliciting sensible priors (or fixed values) for `alpha` or `discount` under the `"IMFA"` and `"IMIFA"` methods for `mcmc_IMIFA`.

Usage

```
G_expected(N,
           alpha,
           discount = 0,
           MPFR = TRUE)

G_variance(N,
           alpha,
           discount = 0,
           MPFR = TRUE)
```

Arguments

`N`	The sample size.
`alpha`	The concentration parameter. Must be specified and must be strictly greater than `-discount`. The case `alpha=0` is accommodated. When `discount` is negative `alpha` must be a positive integer multiple of `abs(discount)`.

`discount` The discount parameter for the Pitman-Yor process. Must be less than 1, but typically lies in the interval [0, 1). Defaults to 0 (i.e. the Dirichlet process). When `discount` is negative `alpha` must be a positive integer multiple of `abs(discount)`.

`MPFR` Logical indicating whether the high-precision libraries `Rmpfr` and `gmp` are invoked, at the expense of run-time. Defaults to `TRUE` and **must** be `TRUE` for `G_expected` when `alpha=0` and `G_variance` when `discount` is non-zero. See `Note`.

Details

All arguments are vectorised. Users can also consult `G_priorDensity` in order to solicit sensible priors.

Value

The expected number of clusters under the specified prior conditions, or the variance of the number of clusters.

Note

`G_variance` requires use of the `Rmpfr` and `gmp` libraries for non-zero `discount` values. `G_expected` requires these libraries only for the `alpha=0` case. Despite the high precision arithmetic used, the functions can still be unstable for small values of `discount`. See the argument `MPFR`.

Author(s)

Keefe Murphy - `<<keefe.murphy@mu.ie>>`

References

De Blasi, P., Favaro, S., Lijoi, A., Mena, R. H., Prunster, I., and Ruggiero, M. (2015) Are Gibbs-type priors the most natural generalization of the Dirichlet process?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2): 212-229.

See Also

`G_priorDensity`, `Rmpfr`

Examples

```
G_expected(N=50, alpha=19.23356, MPFR=FALSE)
G_variance(N=50, alpha=19.23356, MPFR=FALSE)

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

# Examine the growth rate of the DP
DP   <- sapply(c(1, 5, 10), function(i) G_expected(1:200, alpha=i, MPFR=FALSE))
```

```
matplot(DP, type="l", xlab="N", ylab="G")

# Examine the growth rate of the PYP
# PY <- sapply(c(0.25, 0.5, 0.75), function(i) G_expected(1:200, alpha=1, discount=i))
# matplot(PY, type="l", xlab="N", ylab="G")

# Other special cases of the PYP are also facilitated
# G_expected(N=50, alpha=c(27.1401, 0), discount=c(-27.1401/100, 0.8054447))
# G_variance(N=50, alpha=c(27.1401, 0), discount=c(-27.1401/100, 0.8054447))
```

`G_priorDensity` *Plot Pitman-Yor / Dirichlet Process Priors*

Description

Plots the prior distribution of the number of clusters under a Pitman-Yor / Dirichlet 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 (or fixed values) for `alpha` or `discount` under the `"IMFA"` and `"IMIFA"` methods for `mcmc_IMIFA`.

Usage

```
G_priorDensity(N,
               alpha,
               discount = 0,
               show.plot = TRUE,
               type = "h")
```

Arguments

`N`	The sample size.
`alpha`	The concentration parameter. Must be specified and must be strictly greater than `-discount`. The case `alpha=0` is accommodated. When `discount` is negative `alpha` must be a positive integer multiple of `abs(discount)`.
`discount`	The discount parameter for the Pitman-Yor process. Must be less than 1, but typically lies in the interval [0, 1). Defaults to 0 (i.e. the Dirichlet process). When `discount` is negative `alpha` must be a positive integer multiple of `abs(discount)`.
`show.plot`	Logical indicating whether the plot should be displayed (default = `TRUE`).
`type`	The type of plot to be drawn, as per `plot`. Defaults to `"h"`: histogram-like vertical lines.

Details

All arguments are vectorised. Users can also consult `G_expected` and `G_variance` in order to solicit sensible priors.

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

The actual density values are returned invisibly. Therefore, they can be visualised as desired by the user even if `show.plot` is `FALSE`.

Requires use of the `Rmpfr` and `gmp` libraries; may encounter difficulty and slowness for large `N`, especially with non-zero `discount` values. Despite the high precision arithmetic used, the functions can be unstable for small values of `discount`.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

De Blasi, P., Favaro, S., Lijoi, A., Mena, R. H., Prunster, I., and Ruggiero, M. (2015) Are Gibbs-type priors the most natural generalization of the Dirichlet process?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2): 212-229.

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)))

# 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
# alpha    <- c(19.23356, 6.47006, 1)
# discount <- c(0, 0.47002, 0.7300045)
# (PY <- G_priorDensity(N=50, alpha=alpha, discount=discount, type="l"))

#' # Other special cases of the PYP are also facilitated
# G_priorDensity(N=50, alpha=c(alpha, 27.1401, 0),
#                discount=c(discount, -27.1401/100, 0.8054447), type="b")
```

heat_legend *Add a colour key legend to heatmap plots*

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 = NULL,
            breaks = NULL,
            cex.lab = 1,
            ...)
```

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 colour palette used when the heatmap was created. By default, the same viridis default as in mat2cols is used. Will be checked for validity by is.cols.
breaks	Optional argument giving the break-points for the axis labels
cex.lab	Magnification of axis annotation, indicating the amount by which plotting text and symbols should be scaled relative to the default of 1.
...	Catches unused arguments.

Value

Modifies an existing plot by adding a colour-key legend.

See Also

image, plot_cols, mat2cols, is.cols

Examples

```
# Generate a matrix 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, 3.1))

plot_cols(mat2cols(data, col=cols))
heat_legend(data, cols); box(lwd=2)
```

IMIFA_news *Show the NEWS file*

Description

Show the NEWS file of the IMIFA package.

Usage

```
IMIFA_news()
```

Value

The IMIFA NEWS file, provided the session is interactive.

Examples

```
IMIFA_news()
```

is.cols *Check for Valid Colours*

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` *Check Positive-(Semi)definiteness of a matrix*

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

`x`	A matrix, assumed to be real and symmetric.
`tol`	Tolerance for singular values and for absolute eigenvalues - only those with values larger than tol are considered non-zero. (default: tol = `max(dim(x))*max(E)*.Machine$double.eps`, where `E` is the vector of absolute eigenvalues).
`semi`	Logical switch to test for positive-semidefiniteness when `TRUE` or positive-definiteness when `FALSE` (the default).
`make`	Logical switch to return the nearest matrix which satisfies the test - if the test has been passed, this is of course just `x` 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)                                    #FALSE
is.posi_def(x, semi=TRUE)                         #TRUE
```

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

`Ledermann` *Ledermann Bound*

Description

Returns the maximum number of latent factors in a factor analysis model for data of dimension `P` which actually achieves dimension reduction in terms of the number of covariance parameters. 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,
          isotropic = FALSE)
```

Arguments

`P`	Integer number of variables in data set. This argument is vectorised.
`isotropic`	Logical indicating whether uniquenesses are constrained to be isotropic, in which case the bound is simply $P-1$. Defaults to `FALSE`.

Value

The Ledermann bound, a non-negative integer, or a vector of `length(P)` such bounds.

Examples

```
Ledermann(c(25, 50, 100))

data(olive)
Ledermann(ncol(olive[,-c(1,2)]))
```

`mat2cols` *Convert a numeric matrix to colours*

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 = NULL,
         na.col = "#808080FF",
         transparency = 1,
         ...)
```

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 viridis. Will be checked for validity by is.cols.
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). Alternatively, a vector of breakpoints for use with cut.
na.col	Colour to be used to represent missing data. Will be checked for validity by is.cols.
transparency	A factor in [0, 1] modifying the opacity for overplotted lines. Defaults to 1 (i.e. no transparency). Only relevant when cols is not supplied, otherwise the supplied cols must already be adjusted for transparency.
...	Catches unused arguments.

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, cut

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))

# Use plot_cols() to visualise the colours matrix
par(mar=c(5.1, 4.1, 4.1, 3.1))
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     <- cbind(matrix(rnorm(100, sd=c(4,2)),   nr=50, nc=2, byrow=TRUE), 0.1)
mat2     <- cbind(matrix(rnorm(150, sd=c(7,5,3)), nr=50, nc=3, byrow=TRUE), 0.1)
mat3     <- cbind(matrix(rnorm(50,  sd=1),        nr=50, nc=1, byrow=TRUE), 0.1)
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(mat1), cols=cols); box(lwd=2)
plot_cols(mat2cols(mat2, cols=cols)); heat_legend(range(mat2), cols=cols); box(lwd=2)
plot_cols(mat2cols(mat3, cols=cols)); heat_legend(range(mat3), 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 Pitman-Yor / Dirichlet Process Mixture Models (IMFA/IMIFA) are all provided.

Usage

```
mcmc_IMIFA(dat,
           method = c("IMIFA", "IMFA",
                      "OMIFA", "OMFA",
                      "MIFA", "MFA",
                      "IFA", "FA",
                      "classify"),
```

```
            range.G = NULL,
            range.Q = NULL,
            MGP = mgpControl(...),
            BNP = bnpControl(...),
            mixFA = mixfaControl(...),
            alpha = NULL,
            storage = storeControl(...),
            ...)

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

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

Arguments

dat	A matrix or data frame such that rows correspond to observations (N) and columns correspond to variables (P). Non-numeric variables will be discarded if they are explicitly coded as factors or ordinal factors; otherwise they will be treated as though they were continuous. Rows with missing entries will be also be automatically removed.
method	An acronym for the type of model to fit where: "FA" Factor Analysis "IFA" Infinite 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 In principle, of course, one could overfit the "MFA" or "MIFA" models, but it is recommend to use the corresponding model options which begin with 'O' instead. Note that the "classify" method is not yet implemented.
range.G	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 (and must be!) 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 conservatively high number of clusters with which the chain is to be initialised (default = max(25,ceiling(3 * log(N))) for large N, or min(N-1,ceiling(3 * log(N))) for small N<=50). For the "OMFA", and "OMIFA" models this upper limit remains fixed for the entire length of the chain; the upper limit for the for the "IMFA" and "IMIFA" models

can be specified via `trunc.G` (see `bnpControl`), which must satisfy `range.G <= trunc.G < N`.

If `length(range.G) * length(range.Q)` is large, consider not storing unnecessary parameters (via `storeControl`), or breaking up the range of models to be explored into chunks and sending each chunk to `get_IMIFA_results` separately.

`range.Q` 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 `round(3 * log(P))`.

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 `range.Q=0` corresponds to a vanilla Pitman-Yor / Dirichlet Process Mixture Model.

If `length(range.G) * length(range.Q)` is large, consider not storing unnecessary parameters (via `storeControl`), or breaking up the range of models to be explored into chunks and sending each chunk to `get_IMIFA_results`.

See `Ledermann` for bounds on `range.Q`; this is useful in both the finite factor and infinite factor settings, as one may wish to ensure the fixed number of factors, or upper limits on the number of factors, respectively, respects this bound to yield indentifiable solutions, particularly in low-dimensional settings.

`MGP` A list of arguments pertaining to the multiplicative gamma process (MGP) shrinkage prior and adaptive Gibbs sampler (AGS). For use with the infinite factor models `"IFA"`, `"MIFA"`, `"OMIFA"`, and `"IMIFA"` only. Defaults are set by a call to `mgpControl`, with further checking of validity by `MGP_check` (though arguments can also be supplied here directly).

`BNP` A list of arguments pertaining to the Bayesian Nonparametric Pitman-Yor / Dirichlet process priors, for use with the infinite mixture models `"IMFA"` and `"IMIFA"`, or select arguments related to the Dirichlet concentration parameter for the overfitted mixtures `"OMFA"` and `"OMIFA"`. Defaults are set by a call to `bnpControl` (though arguments can also be supplied here directly).

`mixFA` A list of arguments pertaining to *all other* aspects of model fitting, e.g. MCMC settings, cluster initialisation, and hyperparameters common to every `method` in the `IMIFA` family. Defaults are set by a call to `mixfaControl` (though arguments can also be supplied here directly).

`alpha` Depending on the method employed, either the hyperparameter of the Dirichlet prior for the cluster mixing proportions, or the Pitman-Yor / Dirichlet process concentration parameter. Defaults to `1` for the finite mixture models `"MFA"` and `"MIFA"`, and must be a strictly positive scalar. Not relevant for the `"FA"` and `"IFA"` methods.

Under the `"IMFA"` and `"IMIFA"` models: `alpha` defaults to a simulation from the prior if `learn.alpha` is `TRUE`, otherwise `alpha` *must* be specified. Must be positive, unless non-zero `discount` is supplied or `learn.d=TRUE` (the default), in which case it must be greater than `-discount`. Under certain conditions, `alpha` can remain fixed at `0` (see `bnpControl`). Additionally,

when `discount` is negative, `alpha` must be a positive integer multiple of `abs(discount)` (default=`range.G * abs(discount)`).

Under the `"OMFA"` and `"OMIFA"` models: `alpha` defaults to a simulation from the prior if `learn.alpha` is `TRUE`, otherwise `alpha` defaults to `0.5/range.G`. If supplied, `alpha` must be positive, and you are supplying the numerator of `alpha/range.G`.
If `alpha` remains fixed (i.e. `learn.alpha=FALSE`), `alpha` should be less than half the dimension (per cluster!) 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: `PGMM_dfree` and Rousseau and Mengersen (2011)].

See `bnpControl` for further details of specifying `alpha` or specifying a prior for `alpha` under the `"IMFA"`, `"IMIFA"`, `"OMFA"`, or `"OMIFA"` methods.

`storage` A vector of named logical indicators governing storage of parameters of interest for all models in the IMIFA family. Defaults are set by a call to `storeControl`. It may be useful not to store certain parameters if memory is an issue.

`...` An alternative means of passing control parameters directly via the named arguments of `mixfaControl`, `mgpControl`, `bnpControl`, and `storeControl`. Do not pass the output from calls to those functions here!

`x, object` Object of class `"IMIFA"`, for the `print.IMIFA` and `summary.IMIFA` functions, respectively.

Details

Creates a raw object of class `"IMIFA"` from which the optimal/modal model can be extracted by `get_IMIFA_results`. Dedicated `print` and `summary` functions exist for objects of class `"IMIFA"`.

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 are accessible via subsetting, where `x` is of the following 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.

Note

Further control over the specification of advanced function arguments can be obtained with recourse to the following functions:

- `mgpControl` - Supply arguments (with defaults) pertaining to the multiplicative gamma process (MGP) shrinkage prior and adaptive Gibbs sampler (AGS). For use with the infinite factor models `"IFA"`, `"MIFA"`, `"OMIFA"`, and `"IMIFA"` only.
- `bnpControl` - Supply arguments (with defaults) pertaining to the Bayesian Nonparametric Pitman-Yor / Dirichlet process priors, for use with the infinite mixture models `"IMFA"` and `"IMIFA"`. Certain arguments related to the Dirichlet concentration parameter for the overfitted mixtures `"OMFA"` and `"OMIFA"` can be supplied in this manner also.

- `mixfaControl` - Supply arguments (with defaults) pertaining to *all other* aspects of model fitting (e.g. MCMC settings, cluster initialisation, and hyperparameters common to every `method` in the `IMIFA` family.
- `storeControl` - Supply logical indicators governing storage of parameters of interest for all models in the IMIFA family. It may be useful not to store certain parameters if memory is an issue (e.g. for large data sets or for a large number of MCMC iterations after burnin and thinning).

Note however that the named arguments of these functions can also be supplied directly. Parameter starting values are obtained by simulation from the relevant prior distribution specified in these control functions, though initial means and mixing proportions are computed empirically.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

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.

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.

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

See Also

`get_IMIFA_results`, `mixfaControl`, `mgpControl`, `bnpControl`, `storeControl`, `Ledermann`

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.
# Control the balance between the DP and PY priors using the kappa parameter.
# simPY    <- mcmc_IMIFA(olive, method="IMIFA", kappa=0.75)
# 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.
# Don't store the scores. Accept all other defaults.
# simMFA   <- mcmc_IMIFA(olive, method="MFA", n.iters=10000, range.G=3:6, range.Q=0:3,
#                        score.switch=FALSE, 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.
# Enforce additional shrinkage using alpha.d1, alpha.d2, prop, and eps (via mgpControl()).
# simIFA   <- mcmc_IMIFA(olive, method="IFA", n.iters=10000, range.Q=4, scaling="pareto",
#                        alpha.d1=2.5, alpha.d2=4, prop=0.6, eps=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,
#                        phi.hyper=c(2, 1), alpha=0.8, uni.type="single")
```

`mgpControl` *Control settings for the MGP prior and AGS for infinite factor models*

Description

Supplies a list of arguments for use in `mcmc_IMIFA` pertaining to the use of the multiplicative gamma process (MGP) shrinkage prior and adaptive Gibbs sampler (AGS) for use with the infinite factor models `"IFA"`, `"MIFA"`, `"OMIFA"`, and `"IMIFA"`.

Usage

```
mgpControl(alpha.d1 = 2.1,
           alpha.d2 = 3.1,
           phi.hyper = c(3, 2),
           sigma.hyper = c(3, 2),
           prop = 0.7,
           eps = 0.1,
           adapt = TRUE,
           forceQg = FALSE,
           cluster.shrink = TRUE,
           b0 = 0.1,
           b1 = 5e-05,
           beta.d1 = 1,
           beta.d2 = 1,
           start.AGS = 0L,
           stop.AGS = Inf,
```

```
            delta0g = FALSE,
            ...)
```

Arguments

alpha.d1	Shape hyperparameter of the column shrinkage on the first column of the loadings according to the MGP shrinkage prior. Passed to MGP_check to ensure validity. Defaults to 2.1.
alpha.d2	Shape hyperparameter of the column shrinkage on the subsequent columns of the loadings according to the MGP shrinkage prior. Passed to MGP_check to ensure validity. Defaults to 3.1.
phi.hyper	A vector of length 2 giving the shape and rate hyperparameters for the gamma prior on the local shrinkage parameters. Passed to MGP_check to ensure validity. Defaults to c(3,2). It is suggested that the rate be <= shape minus 1 to induce local shrinkage, though the cumulative shrinkage property is unaffected by these hyperparameters. Excessively small values may lead to critical numerical issues and should thus be avoided; indeed it is *suggested* that the shape be >=1.
sigma.hyper	A vector of length 2 giving the shape and rate hyperparameters for the gamma prior on the cluster shrinkage parameters. Passed to MGP_check to ensure validity. Defaults to c(3,2). Again, it is *suggested* that the shape be >= 1. Only relevant for the "IMIFA", "OMIFA", and "MIFA" methods when isTRUE(cluster.shrink).
prop	Proportion of loadings elements within the neighbourhood eps of zero necessary to consider a loadings column redundant. Defaults to floor(0.7 * P)/P, where P is the number of variables in the data set. However, if the data set is univariate or bivariate, the default is 0.5 (see Note).
eps	Neighbourhood epsilon of zero within which a loadings entry is considered negligible according to prop. Defaults to 0.1. Must be positive.
adapt	A logical value indicating whether adaptation of the number of cluster-specific factors is to take place when the MGP prior is employed. Defaults to TRUE. Specifying FALSE and supplying range.Q within mcmc_IMIFA provides a means to either approximate the infinite factor model with a fixed high truncation level, or to use the MGP prior in a finite factor context, however this is NOT recommended for the "OMIFA" and "IMIFA" methods.
forceQg	A logical indicating whether the upper limit on the number of cluster-specific factors Q is also cluster-specific. Defaults to FALSE: when TRUE, the number of factors in each cluster is kept below the number of observations in each cluster, in addition to the bound defined by range.Q. Only relevant for the "IMIFA", "OMIFA", and "MIFA" methods, and only invoked when adapt is TRUE. May be useful for low-dimensional data sets for which identifiable solutions are desired.
cluster.shrink	A logical value indicating whether to place the prior specified by sigma.hyper on the cluster shrinkage parameters. Defaults to TRUE. Specifying FALSE is equivalent to fixing all cluster shrinkage parameters to 1. Only relevant for the "IMIFA", "OMIFA", and "MIFA" methods. If invoked, the posterior mean cluster shrinkage factors will be reported.
b0, b1	Intercept & slope parameters for the exponentially decaying adaptation probability:

p(iter) = 1/exp(b0 + b1 * (iter -start.AGS)).

Defaults to 0.1 & 0.00005, respectively. Must be non-negative and strictly positive, respectively, to ensure diminishing adaptation.

beta.d1 — Rate hyperparameter of the column shrinkage on the first column of the loadings according to the MGP shrinkage prior. Passed to MGP_check to ensure validity. Defaults to 1.

beta.d2 — Rate hyperparameter of the column shrinkage on the subsequent columns of the loadings according to the MGP shrinkage prior. Passed to MGP_check to ensure validity. Defaults to 1.

start.AGS — The iteration at which adaptation under the AGS is to begin. Defaults to burnin for the "IFA" and "MIFA" methods, defaults to 0 for the "OMIFA" and "IMIFA" methods, and defaults to 0 for all methods if the data set is univariate or bivariate. Cannot exceed burnin.

stop.AGS — The iteration at which adaptation under the AGS is to stop completely. Defaults to Inf, such that the AGS is never explicitly forced to stop (thereby overriding the diminishing adaptation probability after stop.AGS). Must be greater than start.AGS. The diminishing adaptation probability prior to stop.AGS is still governed by the arguments b0 and b1.

delta0g — Logical indicating whether the alpha.d1 and alpha.d2 hyperparameters can be cluster-specific. Defaults to FALSE. Only relevant for the "MIFA" method and only allowed when z.list is supplied within mcmc_IMIFA.

... — Catches unused arguments.

Value

A named list in which the names are the names of the arguments related to the MGP and AGS and the values are the values supplied to the arguments.

Note

Certain supplied arguments will be subject to further checks by MGP_check to ensure the cumulative shrinkage property of the MGP prior holds according to the given parameterisation.

The adaptive Gibbs sampler (AGS) monitors the prop of loadings elements within the neighbourhood eps of 0 and discards columns or simulates new columns on this basis. However, if at any stage the number of group-specific latent factors reaches zero, the decision to add columns is instead based on a simple binary trial with probability 1-prop, as there are no loadings entries to monitor.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

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

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

See Also

mcmc_IMIFA, MGP_check, mixfaControl, bnpControl, storeControl

Examples

```
mgpctrl <- mgpControl(phi.hyper=c(2.5, 1), eps=1e-02)

# data(olive)
# sim   <- mcmc_IMIFA(olive, "IMIFA", n.iters=5000, MGP=mgpctrl)

# Alternatively specify these arguments directly
# sim   <- mcmc_IMIFA(olive, "IMIFA", n.iters=5000, phi.hyper=c(2.5, 1), eps=1e-02)
```

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 (in expectation) holds, i.e. checks whether growing mass is assigned to small neighbourhoods of zero as the column index increases.

Usage

```
MGP_check(ad1,
          ad2,
          Q = 3L,
          phi.shape = NULL,
          phi.rate = NULL,
          sigma.shape = NULL,
          sigma.rate = NULL,
          bd1 = 1,
          bd2 = 1,
          inverse = TRUE)
```

Arguments

ad1, ad2	Shape hyperparameters for δ_1 and δ_2, respectively.
Q	Number of latent factors. Defaults to 3, which is enough to check if the cumulative shrinkage property holds. Supply Q if the actual *a priori* expected shrinkage factors are of interest.

`phi.shape, phi.rate`
: The shape and rate hyperparameters for the gamma prior on the local shrinkage parameters. Not necessary for checking if the cumulative shrinkage property holds, but worth supplying *both* if the actual *a priori* expected shrinkage factors are of interest. The default value(s) depends on the value of `inverse`, but are chosen in such a way that the local shrinkage has no effect on the expectation unless both are supplied. Cannot be incorporated into the expectation if `phi.shape < 1` and `isTRUE(inverse)`.

`sigma.shape, sigma.rate`
: The shape and rate hyperparameters for the gamma prior on the cluster shrinkage parameters. Not necessary for checking if the cumulative shrinkage property holds, but worth supplying *both* if the actual *a priori* expected shrinkage factors are of interest. The default value(s) depends on the value of `inverse`, but are chosen in such a way that the cluster shrinkage has no effect on the expectation unless both are supplied. Cannot be incorporated into the expectation if `sigma.shape < 1` and `isTRUE(inverse)`.

`bd1, bd2`
: Rate hyperparameters for δ_1 and δ_2, respectively. Both default to 1.

`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.

Details

This is called inside `mcmc_IMIFA` for the `"IFA"`, `"MIFA"`, `"OMIFA"` and `"IMIFA"` methods. This function is vectorised with respect to the arguments `ad1`, `ad2`, `phi.shape`, `phi.rate`, `sigma.shape`, `sigma.rate`, `bd1` and `bd2`.

Value

A list of length 2 containing the following objects:

- **expectation** - The vector (or list of vectors) of actual expected *a priori* shrinkage factors.
- **valid** - A logical (or vector of logicals) indicating whether the cumulative shrinkage property holds (in expectation).

Note

It is *recommended* that `ad2` be moderately large relative to `ad1`, even if `valid` can sometimes be `TRUE` when this is not the case. Similarly, satisfying this condition is no guarantee that `valid` will be `TRUE`. Therefore, a warning is returned if `ad1 <= ad2`, regardless of the value taken by `valid`.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

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

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

See Also

mcmc_IMIFA

Examples

```
# Check if expected shrinkage under the MGP increases with the column index (WRONG approach!).
MGP_check(ad1=1.5, ad2=1.8, Q=10, phi.shape=3, inverse=FALSE)$valid      #TRUE

# Check if the induced IG prior on the MGP column shrinkage parameters
# is stochastically increasing, thereby inducing cumulative shrinkage    (CORRECT approach!).
MGP_check(ad1=1.5, ad2=1.8, Q=10, phi.shape=3, inverse=TRUE)$valid       #FALSE

# 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, phi.shape=2, phi.rate=0.5, inverse=TRUE))
```

---

mixfaControl *Control settings for the IMIFA family of factor analytic mixtures*

Description

Supplies a list of arguments for use in mcmc_IMIFA pertaining to *ALL* methods in the IMIFA family: eg. MCMC settings, cluster initialisation, generic hyperparameters for factor-analytic mixtures, etc.

Usage

```
mixfaControl(n.iters = 25000L,
             burnin = n.iters/5L,
             thinning = 2L,
             centering = TRUE,
             scaling = c("unit", "pareto", "none"),
             uni.type = c("unconstrained", "isotropic",
                          "constrained", "single"),
             psi.alpha = 2.5,
             psi.beta = NULL,
             mu.zero = NULL,
             sigma.mu = 1L,
             prec.mu = 0.01,
             sigma.l = 1L,
```

```
z.init = c("hc", "kmeans", "list", "mclust", "priors"),
z.list = NULL,
equal.pro = FALSE,
uni.prior = c("unconstrained", "isotropic"),
mu0g = FALSE,
psi0g = FALSE,
drop0sd = TRUE,
verbose = interactive(),
...)
```

Arguments

`n.iters`	The number of iterations to run the sampler for. Defaults to 25000.
`burnin`	The number of burn-in iterations for the sampler. Defaults to `n.iters/5`. Note that chains can also be burned in later, using `get_IMIFA_results`.
`thinning`	The thinning interval used in the simulation. Defaults to 2. No thinning corresponds to 1. Note that chains can also be thinned later, using `get_IMIFA_results`.
`centering`	A logical value indicating whether mean centering should be applied to the data, defaulting to `TRUE`.
`scaling`	The scaling to be applied - one of `"unit"`, `"none"` or `"pareto"`. Defaults to `"unit"`.
`uni.type`	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 clusters. The default `"unconstrained"` corresponds to factor analysis (and mixtures thereof), whereas `"isotropic"` corresponds to probabilistic principal components analysers (and mixtures thereof). Constraints *may* 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: $\Psi_g = \psi\mathcal{I}_p$. `"constrained"` (**UCU**) - variable-specific, equal across clusters: $\Psi_g = \Psi$. `"single"` (**UCC**) - single value equal across clusters and variables: $\Psi_g = \psi\mathcal{I}_p$. The first letter **U** here corresponds to constraints on loadings (not yet implemented), the second letter corresponds to uniquenesses constrained/unconstrained across clusters, and the third letter corresponds to the isotropic constraint on the uniquenesses. 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"`.
`psi.alpha`	The shape of the inverse gamma prior on the uniquenesses. Defaults to 2.5. Must be greater than 1 if `psi.beta` is *not* supplied. Otherwise be warned that values less than or equal to 1 may not bound uniquenesses sufficiently far away from 0, and the algorithm may therefore terminate. Also, excessively small values may lead to critical numerical issues and should thus be avoided.

psi.beta — The scale of the inverse gamma prior on the uniquenesses. Can be either a single parameter, a vector of variable specific scales, or (if psi0g is TRUE) a matrix of variable and cluster-specific scales. If this is not supplied, psi_hyper is invoked to choose sensible values, depending on the value of uni.prior and the data size and dimension, for the "MFA" and "MIFA" models only, the value of psi0g. Excessively small values may lead to critical numerical issues and should thus be avoided.

Note that optional arguments to psi_hyper can be supplied via the ... construct here.

mu.zero — The mean of the prior distribution for the mean parameter. Either a scalar of a vector of appropriate dimension. Defaults to the sample mean of the data.

sigma.mu — The covariance of the prior distribution for the cluster mean parameters. Always assumed to be a diagonal matrix, and set to the identity matrix by default. Can also be a scalar by which the identity is multiplied, a vector of appropriate dimension; if supplied as a matrix, only the diagonal elements will be extracted. Specifying sigma.mu=NULL will use the diagonal entries of the sample covariance matrix: for unit-scaled data this is simply the identity again. See prec.mu for further control over the hypercovariance in the prior for the means.

prec.mu — A scalar controlling the degree of flatness of the prior for the cluster means by scaling sigma.mu (i.e. multiplying every element of sigma.mu by 1/prec.mu). Lower values lead to a more diffuse prior. Defaults to 0.01, such that the prior is relatively non-informative by default. Of course, prec.mu=1 nullifies any effect of this argument. The user can supply a scaled sigma.mu directly, but this argument is especially useful when specifying sigma.mu=NULL, such that the diagonal entries of the sample covariance matrix are used.

sigma.l — A scalar controlling the diagonal covariance of the prior distribution for the loadings. Defaults to 1, i.e. the identity; otherwise a diagonal matrix with non-zero entries all equal to sigma.l Only relevant for the finite factor methods.

z.init — The method used to initialise the cluster labels. Defaults to model-based agglomerative hierarchical clustering via "hc". Other options include "kmeans" (with 10 random starts, by default), Mclust via "mclust", random initialisation via "priors", and a user-supplied "list" (z.list). Not relevant for the "FA" and "IFA" methods. Arguments for the relevant functions can be passed via the ... construct. For "hc", VVV is used by default, unless the data is high-dimensional, in which case the default is EII. The option "priors" may lead to empty components at initialisation, which will return an error.

In any case, unless z.list is explicitly supplied, or verbose is FALSE, the initial cluster sizes will be printed to the screen to alert users to potentially bad initialisiations (e.g. heavily imbalanced initial cluster sizes).

z.list — A user supplied list of cluster labels. Only relevant if z.init == "z.list".

equal.pro — Logical variable indicating whether or not the mixing mixing proportions are to be equal across clusters in the model (default = FALSE). Only relevant for the "MFA" and "MIFA" methods.

uni.prior — A switch indicating whether uniquenesses scale 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"` (though `"isotropic"` is recommended when `N <= P`). Only relevant when `psi.beta` is not supplied and `psi_hyper` is therefore invoked (with optional arguments passable via the `...` construct).

`mu0g` Logical indicating whether the `mu.zero` hyperparameter can be cluster-specific. Defaults to `FALSE`. Only relevant for the `"MFA"` and `"MIFA"` methods when `z.list` is supplied.

`psi0g` Logical indicating whether the `psi.beta` hyperparameter(s) can be cluster-specific. Defaults to `FALSE`. Only relevant for the `"MFA"` and `"MIFA"` methods when `z.list` is supplied, and only allowable when `uni.type` is one of `unconstrained` or `isotropic`.

`drop0sd` Logical indicating whether to drop variables with no standard deviation (defaults to `TRUE`). This is *strongly* recommended, especially a) when `psi.beta` is not supplied &/or `sigma.mu=NULL`, and either/both are therefore estimated using the empirical covariance matrix, &/or b) if some form of posterior predictive checking is subsequently desired when calling `get_IMIFA_results`.

`verbose` 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.

`...` Also catches unused arguments. A number of optional arguments can be also supplied here:

- The additional `psi_hyper` argument `beta0`, especially when `N <= P`.
- Additional arguments to be passed to `hc` (`modelName` & `use` only), to `Mclust` (`modelNames`, and the arguments for `hc` with which `Mclust` is itself initialised - `modelName` & `use`), or to `kmeans` (`iter.max` and `nstart` only), depending on the value of `z.init`.
- Additionally, when `z.init="mclust"`, `criterion` can be passed here (can be `"icl"` or `"bic"`, the default) to control how the optimum `Mclust` model to initialise with is determined.

Value

A named list in which the names are the names of the arguments and the values are the values of the arguments.

Note

Users should be careful to note that data are mean-centered (`centering=TRUE`) and unit-scaled (`scaling="unit"`) by default when supplying other parameters among the list above, especially those related in any way to `psi.hyper`, or to the other control functions `mgpControl` and `bnpControl`.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

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

See Also

mcmc_IMIFA, psi_hyper, hc, kmeans, Mclust, mgpControl, bnpControl, storeControl

Examples

```
mfctrl <- mixfaControl(n.iters=200, prec.mu=1E-03, sigma.mu=NULL,
                       beta0=1, uni.type="constrained")

# data(olive)
# sim  <- mcmc_IMIFA(olive, "IMIFA", mixFA=mfctrl)

# Alternatively specify these arguments directly
# sim  <- mcmc_IMIFA(olive, "IMIFA", n.iters=200, prec.mu=1E-03,
#                    sigma.mu=NULL, beta0=1, uni.type="constrained")
```

---

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, In Martens, H. and Russrum Jr., H. (Eds.), *Food Research and Data Analysis*, Applied Science Publishers, London, pp. 189-214.

Forina, M. and Tiscornia, E. (1982). Pattern recognition methods in the prediction of Italian olive oil origin by their fatty acid content, *Annali di Chimica*, 72:143-155.

Examples

```
data(olive, package="IMIFA")
pairs(olive[,-(1:2)], col=olive$area)
region <- as.numeric(olive$region)
pairs(olive[,-(1:2)],
      col=ifelse(region < 5, 1, ifelse(region < 7, 2, ifelse(region == 9, 4, 3))))
```

---

pareto_scale *Pareto Scaling*

Description

Pareto scaling of a numeric matrix, with or without centering. Observations are scaled by the square-root of their column-wise standard deviations.

Usage

```
pareto_scale(x,
             centering = TRUE)
```

Arguments

x	A numeric matrix-like object.
centering	A logical vector indicating whether centering is to be applied (default=TRUE).

Value

The Pareto scaled version of the matrix x.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

van den Berg, R.A., Hoefsloot, H.C.J, Westerhuis, J.A. and Smilde, A.K. and van der Werf, M.J. (2006) Centering, scaling, and transformations: improving the biological information content of metabolomics data. *BMC Genomics*, 7(142).

Examples

```
dat  <- pareto_scale(olive[,-c(1,2)])
```

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), typically necessary for the penalty term of various model selection criteria.

Usage

```
PGMM_dfree(Q,
           P,
           G = 1L,
           method = c("UUU", "UUC", "UCU", "UCC", "CUU", "CUC",
                      "CCU", "CCC", "CCUU", "UCUU", "CUCU", "UUCU"),
           equal.pro = FALSE)
```

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. Must be a single strictly positive integer.
G	The number of clusters. This defaults to 1. Must be a single strictly positive integer.
method	By default, calculation assumes the UUU model with unconstrained loadings and unconstrained diagonal uniquesseses (i.e. the factor analysis model). The seven other models detailed in McNicholas and Murphy (2008) are given too (of which currently the first four are accommodated within mcmc_IMIFA). The first letter denotes whether loadings are constrained/unconstrained across clusters; the second letter denotes the same for the uniquenesses; the final letter denotes whether uniquenesses are in turn constrained to be isotropic. Finally, the 4 extra 4-letter models from the EPGMM family (McNicholas and Murphy, 2010), are also included.
equal.pro	Logical variable indicating whether or not the mixing mixing proportions are equal across clusters in the model (default = FALSE).

Value

A vector of length length(Q) giving the total number of parameters, including means and mixing proportions, and not only covariance parameters. Set equal.pro to FALSE and subtract G * P from the result to determine the number of covariance parameters only.

Note

This function 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 the `UUU`, `UUC`, `UCU`, and `UCC` covariance structures). The function is vectorised with respect to the argument `Q`.

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 - <<keefe.murphy@mu.ie>>

References

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

McNicholas, P. D. and Murphy, T. B. (2010) Model-Based clustering of microarray expression data via latent Gaussian mixture models, *Bioinformatics*, 26(21): 2705-2712.

See Also

`get_IMIFA_results`, `mcmc_IMIFA`

Examples

```
(UUU <- PGMM_dfree(Q=0:5, P=50, G=3, method="UUU"))
(CCC <- PGMM_dfree(Q=0:5, P=50, G=3, method="CCC", equal.pro=TRUE))
```

---

plot.Results_IMIFA *Plotting output and parameters of inferential interest for IMIFA and related models*

Description

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

Usage

```
## S3 method for class 'Results_IMIFA'
plot(x,
     plot.meth = c("all", "correlation", "density", "errors", "GQ",
                   "means", "parallel.coords", "trace", "zlabels"),
     param = c("means", "scores", "loadings", "uniquenesses",
               "pis", "alpha", "discount"),
     g = NULL,
     mat = TRUE,
     zlabels = NULL,
```

```
heat.map = TRUE,
show.last = FALSE,
palette = NULL,
ind = NULL,
fac = NULL,
by.fac = FALSE,
type = c("h", "n", "p", "l"),
intervals = TRUE,
common = TRUE,
partial = FALSE,
titles = TRUE,
transparency = 0.75,
...)
```

Arguments

x	An object of class "Results_IMIFA" generated by get_IMIFA_results.
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", and "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 over-plotting. Special types of plots which don't require a param are: "GQ" for plotting the posterior summaries of the numbers of clusters/factors, if available. "zlabels" for plotting clustering uncertainties - in four different ways (incl. the posterior confusion matrix) - if clustering has taken place, with or without the clustering labels being supplied via the zlabels argument. If available, the average similarity matrix, reordered according to the MAP labels, is shown as a 5-th plot. "errors" for conducting posterior predictive checking of the appropriateness of the fitted model by visualising the posterior predictive reconstruction error (PPRE) &/or histograms comparing the data to replicate draws from the posterior distribution &/or error metrics quantifying the difference between the estimated and empirical covariance matrices. The type of plot(s) produced depends on how the error.metrics argument was supplied to get_IMIFA_results and what parameters were stored. The argument g can be used to cycle through the available plots in each case. ind can also be used to govern which variable is shown for the 2-nd plot.
param	The parameter of interest for any of the following plot.meth options: all, trace, density, means, correlation. The param must have been stored when mcmc_IMIFA was initially ran. Includes pis for methods where clustering takes place, and allows posterior inference on alpha (for the "IMFA", "IMIFA", "OMFA", and "OMIFA" methods) and discount (for the "IMFA" and "IMIFA" methods). Otherwise "means", "scores", "loadings", and "uniquenesses" can be plotted.

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 plot.meth is GQ, zlabels, or errors in much the same way.
mat	Logical indicating whether a matplot is produced (defaults to TRUE). If given as FALSE, ind is invoked.
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 get_IMIFA_results. Only relevant when plot.meth = "zlabels". When explicitly supplied, misclassified observations are highlighted in the first type of uncertainty plot (otherwise observations whose uncertainty exceed the inverse of the number of clusters are highlighted). For the second type of uncertainty plot, when zlabels are explicitly supplied, the uncertainty of misclassified observations is marked by vertical lines on the profile plot.
heat.map	A logical which controls plotting posterior mean loadings or posterior mean scores as a heatmap, or else as something akin to link{plot(...,type="h")}. 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 mat2cols and plot_cols.
show.last	A logical indicator which defaults to FALSE, but when TRUE replaces any instance of the posterior mean with the last valid sample. Only relevant when param is one of "means" "scores", "loadings", "uniquenesses", or "pis" and plot.meth is one of "all" or "means". Also relevant for "means", "loadings" and "uniquenesses" when plot.meth is "parallel.coords". When TRUE, this has the effect of forcing intervals to be FALSE.
palette	An optional colour palette to be supplied if overwriting the default palette set inside the function by viridis is desired. It makes little sense to a supply a palette when plot.meth="all" and param is one of "scores" or "loadings".
ind	Either a single number indicating which variable to plot when param is one of means or uniquenesses (or plot.meth="errors"), or which cluster to plot if param is pis. 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. Will be recycled to length 2 if necessary. Also governs which two factors are displayed on posterior mean plots of the "scores" when heat.map is FALSE; otherwise only relevant when mat is FALSE.
fac	Optional argument that provides an alternative way to specify ind[2] when mat is FALSE and param 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 ind or fac) when set to TRUE. Otherwise all factor(s) are plotted for a given observation or variable when set to FALSE (the default), again controlled by ind or fac. Only relevant when param is one of scores or loadings.
type	The manner in which the plot is to be drawn, as per the type argument to plot.
intervals	Logical indicating whether credible intervals around the posterior mean(s) are to be plotted when is.element(plot.meth,c("all","means")). Defaults to TRUE, but can only be TRUE when show.last is FALSE.

common	Logical indicating whether plots with plot.meth="means" (or the corresponding plots for plot.meth="all") when param is one of "means", "scores", "loadings", or "uniquenesses" are calibrated to a common scale based on the range of the param parameters across all clusters (defaults to TRUE, and only relevant when there are clusters). Otherwise, the only the range corresponding to the image being plotted is used to determine the scale. Note that this affects the "loadings" and "scores" plots regardless of the value of heat.map. An exception is the "scores" plots when plot.meth="means" and heat.map is FALSE, in which case common defaults to FALSE.
partial	Logical indicating whether plots of type "correlation" use the PACF. The default, FALSE, ensures the ACF is used. Only relevant when plot.meth = "all", otherwise both plots are produced when plot.meth = "correlation".
titles	Logical indicating whether default plot titles are to be used (TRUE), or suppressed (FALSE).
transparency	A factor in [0, 1] modifying the opacity for overplotted lines. Defaults to 0.75, unless semi-transparency is not supported. Only relevant when palette is not supplied, otherwise the supplied palette must already be adjusted for transparency.
...	Other arguments typically passed to plot or the breaks argument to mat2cols and heat_legend when heatmaps are plotted.

Value

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

Note

Supplying the argument zlabels does **not** have the same effect of reordering the sampled parameters as it does if supplied directly to get_IMIFA_results.

When mat is TRUE and by.fac is FALSE (both defaults), the convention for dealing with overplotting for trace and density plots when param is either scores or loadings is to plot the last factor first, such that the first factor appears 'on top'.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

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)
# 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=olive$area)

# Visualise the posterior predictive reconstruction error
# plot(resIMIFA, plot.meth="errors", g=1)

# Compare histograms of the data vs. replicate draw from the posterior for the 1st variable
# plot(resIMIFA, plot.meth="errors", g=2, ind=1)

# Visualise empirical vs. estimated covariance error metrics
# plot(resIMIFA, plot.meth="errors", g=3)

# 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="trace", param="scores")
# plot(resIMIFA, plot.meth="trace", param="scores", by.fac=TRUE)
# plot(resIMIFA, plot.meth="mean", param="loadings", g=1)
# plot(resIMIFA, plot.meth="mean", param="loadings", g=1, heat.map=FALSE)
# plot(resIMIFA, plot.meth="parallel.coords", param="uniquenesses")
# plot(resIMIFA, plot.meth="density", param="pis", intervals=FALSE, partial=TRUE)
# plot(resIMIFA, plot.meth="all", param="alpha")
# plot(resIMIFA, plot.meth="all", param="discount")
```

---

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, rlabels, clabels	Vector of labels for the diagonals, rows, and columns, respectively.
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" type 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))

# Use plot_cols() to visualise the colours matrix
par(mar=c(5.1, 4.1, 4.1, 3.1))
plot_cols(matcol)

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

# Replace colour of exact zero entries:
```

```
# Often important to call mat2cols() first (to include 0 in the cuts),
# then replace relevant entries with NA for plot_cols(), i.e.
mat[2,3] <- 0
matcol2  <- mat2cols(mat, cols=cols)
plot_cols(replace(matcol2, mat == 0, NA), na.col="blue")
heat_legend(mat, cols=cols); box(lwd=2)
```

`post_conf_mat` *Posterior Confusion Matrix*

Description

For a (`N * G`) matrix of posterior cluster membership probabilities, this function creates a (`G * G`) posterior confusion matrix, whose hk-th entry gives the average probability that observations with maximum posterior allocation h will be assigned to cluster k.

Usage

```
post_conf_mat(z, scale = TRUE)
```

Arguments

`z`	A (`N * G`) matrix of posterior cluster membership probabilities whose ig-th entry gives the posterior probability that observation i belongs to cluster g. Entries must be valid probabilities in the interval [0,1]; missing values are not allowed. Otherwise, a list of such matrices can be supplied, where each matrix in the list has the same dimensions.
`scale`	A logical indicator whether the PCM should be rescaled by its row sums. When `TRUE` (the default), the benchmark matrix for comparison is the identity matrix of order `G`, corresponding to a situation with no uncertainty in the clustering. When `FALSE`, the row sums give the number of observations in each cluster.

Value

A (`G * G`) posterior confusion matrix, whose hk-th entry gives the average probability that observations with maximum posterior allocation h will be assigned to cluster k. When `scale=TRUE`, the benchmark matrix for comparison is the identity matrix of order `G`, corresponding to a situation with no uncertainty in the clustering.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Ranciati, S., Vinciotti, V. and Wit, E., (2017) Identifying overlapping terrorist cells from the Noordin Top actor-event network, *to appear*. <arXiv:1710.10319v1>.

See Also

get_IMIFA_results

Examples

```
# data(olive)
# sim  <- mcmc_IMIFA(olive, n.iters=1000)
# res  <- get_IMIFA_results(sim)
# (PCM <- post_conf_mat(res$Clust$post.prob))

# par(mar=c(5.1, 4.1, 4.1, 3.1))
# PCM  <- replace(PCM, PCM == 0, NA)
# plot_cols(mat2cols(PCM, col=heat.colors(30)[30:1], na.col=par()$bg)); box(lwd=2)
# heat_legend(PCM, cols=heat.colors(30)[30:1])
# par(mar=c(5.1, 4.1, 4.1, 2.1))
```

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 Xstar.

Usage

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

Arguments

X	The matrix to be transformed.
Xstar	The target matrix.
translate	Logical value indicating whether X should be translated (defaults to FALSE).
dilate	Logical value indicating whether X should be dilated (defaults to FALSE).
sumsq	Logical value indicating whether the sum of squared differences between X and Xstar should be calculated and returned.

Details

R, tt, and d are chosen so that:

$$d \times \mathbf{XR} + \mathbb{1}\underline{t}^\top \approx X^\star$$

X.new is given by:

$$X_{\text{new}} = d \times \mathbf{XR} + \mathbb{1}\underline{t}^\top$$

Value

A list containing:

X.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 (is isTRUE(dilate)).
ss	The sum of squared differences (if isTRUE(sumsq)).

Note

X is padded out with columns containing 0 if it has fewer columns than Xstar.

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 *Find sensible inverse gamma hyperparameters for variance/uniqueness parameters*

Description

Takes an inverse-Gamma shape hyperparameter, and an inverse covariance matrix (or estimate thereof), and finds data-driven scale hyperparameters in such a way that Heywood problems are avoided for factor analysis or probabilistic principal components analysis (and mixtures thereof).

Usage

```
psi_hyper(shape,
          dat,
          type = c("unconstrained", "isotropic"),
          beta0 = 3,
          ...)
```

Arguments

shape	A positive shape hyperparameter.
dat	The data matrix from which the inverse covariance matrix is estimated. If data are to be centered &/or scaled within mcmc_IMIFA, then dat must also be standardised in the same way.
type	A switch indicating whether a single scale (isotropic) or variable-specific scales (unconstrained) are to be derived. Both options are allowed under models in mcmc_IMIFA with "constrained" or "unconstrained" uniquenesses, but only a single scale can be specified for models with "isotropic" or "single" uniquenesses.
beta0	See Note below. Must be a strictly positive numeric scalar. Defaults to 3, but is only invoked when explicitly supplied or when the number of observations N is not greater than the number of variables P (or when inverting the sample covariance matrix somehow otherwise fails). In some cases, beta0 should not be so small as to render the estimate of the inverse unstable.
...	Catches unused arguments. Advanced users can also supply the sample covariance matrix of dat, if available, through ... via the argument covar.

Details

Constraining uniquenesses to be isotropic provides the link between factor analysis and the probabilistic PCA model. When used in conjunction with mcmc_IMIFA with "isotropic" or "single" uniquenesses, type must be isotropic, but for "unconstrained" or "constrained" uniquenesses, it's possible to specify either a single scale (type="isotropic") or variable-specific scales (type="unconstrained").

Used internally by mcmc_IMIFA when its argument psi_beta is not supplied.

Value

Either a single scale hyperparameter or `ncol(dat)` variable-specific scale hyperparameters.

Note

When `N > P`, where `N` is the number of observations and `P` is the number of variables, the inverse of the sample covariance matrix is used by default.

When `N <= P`, the inverse either does not exist or the estimate thereof is highly unstable. Thus, an estimate of the form $\left(\beta_0 + \frac{N}{2}\right)\left(\beta_0 \mathcal{I}_p + 0.5\sum_{i=1}^{N} x_i x_i^\top\right)^{-1}$ is used instead.

For unstandardised data, the estimate is instead constructed using a standardised version of the data, and the resulting inverse *correlation* matrix estimate is scaled appropriately by the diagonal entries of the sample covariance matrix of the original data.

This estimate can also be used in `N > P` cases by explicitly supplying `beta0`. It will also be used if inverting the sample covariance matrix fails in `N > P` cases.

The optional argument `beta0` can be supplied to `mcmc_IMIFA` via the control function `mixfaControl`.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

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.

Fruwirth-Schnatter, S. and Lopes, H. F. (2018). Sparse Bayesian factor analysis when the number of factors is unknown, *to appear*. <arXiv:1804.04231>.

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`, `mixfaControl`

Examples

```
data(olive)
olive2  <- olive[,-(1:2)]
(scale1 <- psi_hyper(shape=2.5, dat=olive2))

# Try again with scaled data
olive_S <- scale(olive2, center=TRUE, scale=TRUE)

# Use the inverse of the sample covariance matrix
(scale2 <- psi_hyper(shape=2.5, dat=olive_S))
```

```
# Use the estimated inverse covariance matrix
(scale3 <- psi_hyper(shape=2.5, dat=olive_S, beta0=3))

# In the normalised example, the mean uniquenesses (given by scale/(shape - 1)),
# can be interpreted as the prior proportion of the variance that is idiosyncratic
(prop1   <- scale1/(2.5 - 1))
(prop2   <- scale2/(2.5 - 1))
```

`rDirichlet` *Simulate Mixing Proportions from a Dirichlet Distribution*

Description

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

Usage

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

Arguments

`G`	The number of clusters 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, which doesn't favour one component over another. Be warned that this will be recycled if necessary. Larger values have the effect of making the returned samples more equal.
`nn`	A vector giving the number of observations in each of G clusters so that Dirichlet posteriors rather than priors can be sampled from. This defaults to 0, i.e. simulation from the prior. Must be non-negative. 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 few checks take place, in order to speed up repeated calls to the function inside `mcmc_IMIFA`. In particular, `alpha` and `nn` may be invisibly recycled.

While small values of `alpha` have the effect of increasingly concentrating the mass onto fewer components, note that this function may return `NaN` for excessively small values of `alpha`, when `nn=0`; see the details of `rgamma` for small `shape` values.

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)))
(asymmetric <- rDirichlet(G=5, alpha=c(3,4,5,1,2), nn=c(20, 41, 32, 8, 12)))
```

---

scores_MAP *Decompose factor scores by cluster*

Description

Takes posterior summaries of the overall factor scores matrix and returns lists of sub-matrices corresponding to the G-cluster MAP partition.

Usage

```
scores_MAP(res,
           dropQ = FALSE)
```

Arguments

res	An object of class "Results_IMIFA" generated by get_IMIFA_results.
dropQ	A logical indicating whether columns of the factor scores matrix should be dropped such that the number of columns in each sub-matrix corresponds to the cluster-specific number of factors (if the number of factors is indeed cluster-specific). When FALSE (the default), the number of columns instead remains common to all sub-matrices - given by the largest of the cluster-specific numbers of latent factors. Note that this argument is irrelevant (i.e. always FALSE) for the finite factor methods ("FA", "MFA", "OMFA", and "IMFA").

Details

Under the models in the IMIFA family, there exists only one factor scores matrix. For the finite factor methods, this has dimensions N * Q.

For the infinite factor methods ("IFA", "MIFA", "OMIFA", and "IMIFA"), the factor scores matrix has dimensions N * Qmax, where Qmax is the largest of the cluster-specific numbers of latent factors $q_1, \ldots, q_g$. Entries of this matrix thus may have been padded out with zero entries, as appropriate, prior to the Procrustes rotation-based correction applied within get_IMIFA_results (thus now these entries will be near-zero).

In partitioning rows of the factor scores matrix into the same clusters the corresponding observations themselves belong to according to the MAP clustering, the number of columns *may* vary according to the cluster-specific numbers of latent factors (depending on the value of dropQ and the method employed).

Value

For models which achieve clustering, a list of lists (say `x`) decomposing the posterior mean scores (`x$post.eta`), the associated variance estimates (`x$var.eta`) and credible intervals (`x$ci.eta`), and the last valid sample of the scores (`x$last.eta`) into lists of length `G`, corresponding to the MAP clustering, with varying or common numbers of cluster-specific factors (depending on the value of `dropQ` and the method employed).

For models with only one component, or the `"FA"` and `"IFA"` methods, scores cannot be decomposed, and posterior summaries of the scores will be returned unchanged.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

See Also

`get_IMIFA_results`

Examples

```
data(coffee)
sim <- mcmc_IMIFA(coffee, n.iters=1000)
res <- get_IMIFA_results(sim)

# Examine the single posterior mean scores matrix
res$Scores$post.eta

# Decompose into G matrices, common numbers of columns
eta <- scores_MAP(res)
eta$post.eta

# Allow the number of columns be cluster-specific
scores_MAP(res, dropQ=TRUE)$post.eta
```

---

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 = 0,
         param = c("rate", "scale"))
```

Arguments

shape, rate	Shape and rate parameters a and b, respectively, of a Gamma(a, b) distribution. Both must be strictly positive.
shift	Modifier, such that the Gamma distribution has support on (shift, ∞). 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 named vector of length 2, containing the modified shape and rate parameters, respectively.

Note

This function is invoked within mcmc_IMIFA when discount is *fixed* at a non-zero value and learn.alpha=TRUE.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

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)
```

---

show_digit *Show image of grayscale grid*

Description

Plots an image of a grayscale grid representation of a digit.

Usage

```
show_digit(dat,
           col = NULL,
           ...)
```

Arguments

dat	A matrix or data.frame with the same number of rows and columns (or a vector which can be coerced to such a format), representing a grayscale map of a single digit.
col	The colour scale to be used. Defaults to grey(seq(1,0,length = ncol(dat))).
...	Additional arguments to be passed to mat2cols and/or plot_cols (e.g. na.col) when dat is a matrix or image when dat is a vector.

Value

The desired image representation of the digit.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

See Also

USPSdigits, show_IMIFA_digit, mat2cols, plot_cols

Examples

```
data(USPSdigits)

# Plot the first digit
show_digit(USPSdigits$train[1,-1])

# Visualise the overall mean
show_digit(colMeans(USPSdigits$train[,-1]))
```

---

show_IMIFA_digit *Plot the posterior mean image*

Description

Plots the posterior mean of a given cluster from an "IMIFA"-related model fit to a digit data set in the form of a square grayscale grid.

Usage

```
show_IMIFA_digit(res,
                 G = 1,
                 what = c("mean", "last"),
                 dat = NULL,
                 ind = NULL,
                 ...)
```

Arguments

res	An object of class "Results_IMIFA" generated by get_IMIFA_results.
G	The index of the cluster for which the posterior mean digit is to be represented.
what	A switch controlling whether the "mean" or "last" valid sample is to be plotted.
dat	The full grayscale grid data set (prior to centering and scaling). Necessary when ind is supplied or if pixels with standard deviation of 0 exist in the data set (which will have been automatically removed by mcmc_IMIFA).

ind	The index of columns of dat which were discarded prior to fitting the "IMIFA"-related model via mcmc_IMIFA. Can be a vector of column indices of dat or an equivalent vector of logicals. The discarded pixels are replaced by the column-means corresponding to ind among images assigned to the given cluster G.
...	Additional arguments to be passed, via show_digit, to mat2cols and/or plot_cols.

Details

This function is a wrapper to show_digit which supplies the posterior mean digit of a given cluster from a "IMIFA" model.

Value

The desired image representation of the posterior mean digit (or the last valid sample) from the desired cluster.

Note

Note that both centering and scaling of the original data prior to modelling is accounted for in reconstructing the means, but dat, if necessary, must be the raw data prior to pre-processing.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

See Also

USPSdigits, show_digit, get_IMIFA_results, mcmc_IMIFA, mat2cols, plot_cols

Examples

```
# Load the USPS data and discard peripheral digits
data(USPSdigits)
ylab  <- USPSdigits$train[,1]
train <- USPSdigits$train[,-1]
ind   <- apply(train, 2, sd) > 0.7
dat   <- train[,ind]

# Fit an IMIFA model (warning: quite slow!)
# sim <- mcmc_IMIFA(dat, n.iters=1000, prec.mu=1e-03, z.init="kmeans",
#                   centering=FALSE, scaling="none")
# res <- get_IMIFA_results(sim, zlabels=ylab)

# Examine the posterior mean image of the first two clusters
# show_IMIFA_digit(res, dat=train, ind=ind)
# show_IMIFA_digit(res, dat=train, ind=ind, G=2)
```

`sim_IMIFA` *Simulate Data from a Mixture of Factor Analysers Structure*

Description

Functions to simulate data of any size and dimension from a (infinite) mixture of (infinite) factor analysers parameterisation or fitted object.

Usage

```
sim_IMIFA_data(N = 300L,
               G = 3L,
               P = 50L,
               Q = rep(floor(log(P)), G),
               pis = rep(1/G, G),
               mu = NULL,
               psi = NULL,
               loadings = NULL,
               scores = NULL,
               nn = NULL,
               loc.diff = 2,
               non.zero = P,
               forceQg = TRUE,
               method = c("conditional", "marginal"))

sim_IMIFA_model(res,
                method = c("conditional", "marginal"))
```

Arguments

`N, G, P`	Desired overall number of observations, number of clusters, and number of variables in the simulated data set. All must be 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 `floor(log(P))` in each cluster. Should be less than the associated `Ledermann` bound and the number of observations in the corresponding cluster. The argument `forceQg` can be used to enforce this upper limit.
`pis`	Mixing proportions of the clusters in the dataset if `G` > 1. Must sum to 1. Defaults to `rep(1/G,G)`.
`mu`	True values of the mean parameters, either as a single value, a vector of length `G`, a vector of length `P`, or a `G * P` matrix. If `mu` is missing, `loc.diff` is invoked to simulate distinct means for each cluster by default.
`psi`	True values of uniqueness parameters, either as a single value, a vector of length `G`, a vector of length `P`, or a `G * P` matrix. As such the user can specify uniquenesses as a diagonal or isotropic matrix, and further constrain uniquenesses

across clusters if desired. If `psi` is missing, uniquenesses are simulated via `1/rgamma(P,2,1)` within each cluster by default.

`loadings` True values of the loadings matrix/matrices. Must be supplied in the form of a list of numeric matrices when `G > 1`, otherwise a single matrix. Matrices must contain `P` rows and the number of columns must correspond to the values in `Q`. If `loadings` are not supplied, such matrices are populated with standard normal random variates by default (see `non.zero`).

`scores` True values of the latent factor scores, as a `N * max(Q)` numeric matrix. If `scores` are not supplied, such a matrix is populated with standard normal random variates by default. Only relevant when `method="conditional"`.

`nn` An alternative way to specify the size of each cluster, by giving the exact number of observations in each cluster 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. Only relevant if `mu` is NOT supplied. Defaults to `2`.

More specifically, `loc.diff` (if invoked) is invoked as follows: means are simulated with the vector of cluster-specific hypermeans given by:

`scale(1:G,center=TRUE,scale=FALSE) * loc.diff`.

`non.zero` Controls the number of non-zero entries in each loadings column (per cluster) **only** when `loadings` is not explicitly supplied. Values must be integers in the interval `[1,P]`. Defaults to `P`. The positions of the zeros are randomised, and non-zero entries are drawn from a standard normal.

Must be given as a list of length `G` of vectors of length corresponding to `Q` when `G>1`. Can be given either as such a list or simply a vector of length `Q` when `G=1`. Alternatively, a single integer can be supplied, common across all loadings columns across all clusters. In any case, `non.zero` will be affected by `forceQg=TRUE` by default (see below).

`forceQg` A logical indicating whether the upper limit on the number of cluster-specific factors `Q` is enforced. Defaults to `TRUE` for `sim_IMIFA_data`, but is always `FALSE` for `sim_IMIFA_model`. Note that when `forceQg=TRUE` is invoked, `non.zero` (see above) is also affected.

`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.

`res` An object of class `"Results_IMIFA"` generated by `get_IMIFA_results`.

Details

`sim_IMIFA_model` is a simple wrapper to `sim_IMIFA_data` which uses the estimated parameters of a fitted IMIFA related model, as generated by `get_IMIFA_results`. The necessary parameters must have been originally stored via `storeControl` in the creation of `res`.

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 as attributes.

Note

N, G, P & Q will **NOT** be inferred from the supplied parameters pis, mu, psi, loadings, scores & nn - rather, the parameters' length/dimensions must adhere to the supplied values of N, G, P & Q.

Missing values are not allowed in any of pis, mu, psi, loadings, scores & nn.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Murphy, K., Viroli, C., and Gormley, I. C. (2020) Infinite mixtures of infinite factor analysers, *Bayesian Analysis*, 15(3): 937-963. <doi:10.1214/19-BA1179>.

See Also

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

get_IMIFA_results for generating input for sim_IMIFA_model.

Ledermann for details on the upper-bound for Q. Note that this function accounts for isotropic uniquenesses, if psi is supplied in that manner, in computing this bound.

Examples

```
# Simulate 100 observations from 3 balanced clusters with cluster-specific numbers of latent factors
# Specify isotropic uniquenesses within each cluster
# Supply cluster means directly
sim_data  <- sim_IMIFA_data(N=100, G=3, P=20, Q=c(2, 2, 5), psi=1:3,
                            mu=matrix(rnorm(60, -2 + 1:3, 1), nrow=20, ncol=3, byrow=TRUE))
names(attributes(sim_data))
labels    <- attr(sim_data, "Labels")

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

# Examine the overlap with a pairs plot of 5 randomly chosen variables
pairs(sim_data[,sample(1:20, 5)], col=labels)

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

# Simulate from this model
# res     <- get_IMIFA_results(tmp, zlabels=labels)
# sim_mod <- sim_IMIFA_model(res)
```

`storeControl` *Set storage values for use with the IMIFA family of models*

Description

Supplies a list of values for logical switches indicating whether parameters of interest (means, scores, loadings, uniquenesses, and mixing proportions) should be stored when running models from the IMIFA family via `mcmc_IMIFA`. It may be useful not to store certain parameters if memory is an issue.

Usage

```
storeControl(mu.switch = TRUE,
             score.switch = TRUE,
             load.switch = TRUE,
             psi.switch = TRUE,
             pi.switch = TRUE,
             ...)
```

Arguments

`mu.switch`	Logical indicating whether the means are to be stored (defaults to `TRUE`).
`score.switch`	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` inside `mcmc_IMIFA` when `length(range.G) * length(range.Q) > 10`, otherwise the default is `TRUE`. For the `"MIFA"`, `"OMIFA"`, and `"IMIFA"` methods, setting this switch to `FALSE` also offers a slight speed-up. Unlike other parameters, the scores need not be stored for posterior predictive checking (see Note below).
`load.switch`	Logical indicating whether the factor loadings are to be stored (defaults to `TRUE`).
`psi.switch`	Logical indicating whether the uniquenesses are to be stored (defaults to `TRUE`).
`pi.switch`	Logical indicating whether the mixing proportions are to be stored (defaults to `TRUE`).
`...`	Catches unused arguments.

Details

`storeControl` is provided for assigning values for IMIFA models within `mcmc_IMIFA`. It may be useful not to store certain parameters if memory is an issue (e.g. for large data sets or for a large number of MCMC iterations after burnin and thinning).

Value

A named vector in which the names are the names of the storage switches and the values are logicals indicating whether that parameter is to be stored. The list also contains as an attribute a logical for each switch indicating whether it was actually supplied (`TRUE`) or the default was accepted (`FALSE`).

Note

Posterior inference and plotting won't be possible for parameters not stored.

Non-storage of parameters will almost surely prohibit the computation of posterior predictive checking error metrics within `get_IMIFA_results` also. In particular, if such error metrics are desired, `mu.switch` and `psi.switch` must be `TRUE` for all but the `"FA"` and `"IFA"` models, `load.switch` must be `TRUE` for all but the entirely zero-factor models, and `pi.switch` must be `TRUE` for models with clustering structure and unequal mixing proportions for all but the PPRE metric. `score.switch=TRUE` is not required for any posterior predictive checking.

Finally, if loadings are not stored but scores are, caution is advised when examining posterior scores as Procrustes rotation will not occur within `get_IMIFA_results`.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

See Also

`mcmc_IMIFA`, `get_IMIFA_results`, `mixfaControl`, `mgpControl`, `bnpControl`

Examples

```
stctrl <- storeControl(score.switch=FALSE)

# data(olive)
# sim  <- mcmc_IMIFA(olive, "IMIFA", n.iters=5000, storage=stctrl)

# Alternatively specify these arguments directly
# sim  <- mcmc_IMIFA(olive, "IMIFA", n.iters=5000, score.switch=FALSE)
```

USPSdigits *USPS handwritten digits*

Description

Training and test sets for the United States Postal Service (USPS) handwritten digits data, with 8-bit 16x16 grayscale grid representations of image scans of the digits "0" through "9".

Usage

```
data(USPSdigits)
```

Format

A list of length 2 with the following elements, each one a `data.frame`:

`train` The training set of 7,291 digits.

`test` The test set of 2,007 digits.

Each `data.frame` contains the known digit labels in its first column.

The remaining 256 columns give the concatenation of the 16x16 grid.

Pixels are scaled such that [-1,1] corresponds to [white,black].

References

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Spring New York Inc., 2nd edition.

See Also

`show_digit`, `show_IMIFA_digit`

Examples

```
# Load the data and record the labels
data(USPSdigits, package="IMIFA")
ylab  <- USPSdigits$train[,1]
train <- USPSdigits$train[,-1]

# Examine the effect of discarding peripheral pixels
SDs   <- apply(train, 2, sd)
ind   <- SDs > 0.7
dat   <- train[,ind]

hist(SDs, breaks=200, xlim=c(0, 1))
rect(0.7, 0, 1, 12, col=2, density=25)

show_digit(ind) # retained pixels are shown in black
```

Zsimilarity *Summarise MCMC samples of clustering labels with a similarity matrix and find the 'average' clustering*

Description

This function takes a Monte Carlo sample of cluster labels, computes an average similarity matrix and returns the clustering with minimum mean squared error to this average. The `mcclust` package **must** be loaded.

Usage

```
Zsimilarity(zs)
```

Arguments

zs	A matrix containing samples of clustering labels where the columns correspond to the number of observations (N) and the rows correspond to the number of iterations (M).

Details

This function 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, desirable features when summarising partitions of Bayesian nonparametric models such as IMIFA. As a summary of the posterior clustering, the clustering with minimum mean squared error to this 'average' clustering is reported.

A heatmap of `z.sim` may provide a useful visualisation, if appropriately ordered. 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. Alternatively, `hc` could be used.

Value

A list containing three elements:

z.avg	The 'average' clustering, with minimum squared distance to `z.sim`.
z.sim	The N x N similarity matrix, in a sparse format (see `simple_triplet_matrix`).
MSE.z	A vector of length M recording the MSEs between each clustering and the 'average' clustering.

Note

The `mcclust` package **must** be loaded.

This is liable to take quite some time to run, especially if the number of observations &/or number of iterations is large. Depending on how distinct the clusters are, `z.sim` may be stored better in a non-sparse format. This function can optionally be called inside `get_IMIFA_results`.

Author(s)

Keefe Murphy - <<keefe.murphy@mu.ie>>

References

Carmona, C., Nieto-barajas, L. and Canale, A. (2018) Model based approach for household clustering with mixed scale variables. *Advances in Data Analysis and Classification*, 13(2): 559-583.

See Also

`get_IMIFA_results`, `simple_triplet_matrix`, `hclust`, `hc`, `comp.psm`, `cltoSim`

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$z.sim)
# z.col  <- mat2cols(z.sim, cols=heat.colors(30)[30:1])
# z.col[z.sim == 0] <- NA
# plot_cols(z.col, na.col=par()$bg); box(lwd=2)

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

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

Index