Package ‘GMCM’

January 20, 2025

Type Package

Title Fast Estimation of Gaussian Mixture Copula Models

Description Unsupervised Clustering and Meta-analysis using Gaussian Mixture Copula Models.

Version 1.4

Maintainer Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

URL https://github.com/AEBilgrau/GMCM

BugReports https://github.com/AEBilgrau/GMCM/issues

License GPL (>= 2)

KeepSource yes

Imports Rcpp (>= 0.10.6), ellipse

LinkingTo Rcpp, RcppArmadillo

Suggests idr, Hmisc, RColorBrewer, foreach, jpeg, testthat (>= 0.3), knitr, rmarkdown, shiny, shinydashboard, shinyBS, rhandsontable, DT

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation yes

Author Anders Ellern Bilgrau [aut, cre, cph] (<https://orcid.org/0000-0001-9875-2902>), Poul Svante Eriksen [ths, ctb] (<https://orcid.org/0000-0001-9192-1814>), Martin Boegsted [ths, ctb]

Repository CRAN

Date/Publication 2019-11-05 22:10:02 UTC

Contents

`GMCM-package` *Fast optimization of Gaussian Mixture Copula Models*

Description

Gaussian mixture copula models (GMCM) are a flexible class of statistical models which can be used for unsupervised clustering, meta analysis, and many other things. In meta analysis, GMCMs can be used to quantify and identify which features which have been reproduced across multiple experiments. This package provides a fast and general implementation of GMCM cluster analysis and serves as an improvement and extension of the features available in the `idr` package.

Details

If the meta analysis of Li et al. (2011) is to be performed, the function `fit.meta.GMCM` is used to identify the maximum likelihood estimate of the special Gaussian mixture copula model (GMCM) defined by Li et al. (2011). The function `get.IDR` computes the local and adjusted Irreproducible Discovery Rates defined by Li et al. (2011) to determine the level of reproducibility.

Tewari et. al. (2011) proposed using GMCMs as an general unsupervised clustering tool. If such a general unsupervised clustering is needed, like above, the function `fit.full.GMCM` computes the maximum likelihood estimate of the general GMCM. The function `get.prob` is used to estimate the class membership probabilities of each observation.

`SimulateGMCMData` provide easy simulation from the GMCMs.

Author(s)

Anders Ellern Bilgrau, Martin Boegsted, Poul Svante Eriksen

Maintainer: Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Anders Ellern Bilgrau, Poul Svante Eriksen, Jakob Gulddahl Rasmussen, Hans Erik Johnsen, Karen Dybkaer, Martin Boegsted (2016). GMCM: Unsupervised Clustering and Meta-Analysis Using Gaussian Mixture Copula Models. Journal of Statistical Software, 70(2), 1-23. doi:10.18637/jss.v070.i02

Li, Q., Brown, J. B. J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. The Annals of Applied Statistics, 5(3), 1752-1779. doi:10.1214/11-AOAS466

Tewari, A., Giering, M. J., & Raghunathan, A. (2011). Parametric Characterization of Multimodal Distributions with Non-gaussian Modes. 2011 IEEE 11th International Conference on Data Mining Workshops, 286-292. doi:10.1109/ICDMW.2011.135

See Also

Core user functions: `fit.meta.GMCM`, `fit.full.GMCM`, `get.IDR`, `get.prob`, `SimulateGMCMData`, `SimulateGMMData`, `rtheta`, `Uhat`, `choose.theta`, `full2meta`, `meta2full`

Package by Li et. al. (2011): `idr`.

Examples

```
# Loading data
data(u133VsExon)

# Subsetting data to reduce computation time
u133VsExon <- u133VsExon[1:5000, ]

# Ranking and scaling,
# Remember large values should be critical to the null!
uhat <- Uhat(1 - u133VsExon)

# Visualizing P-values and the ranked and scaled P-values
## Not run:
par(mfrow = c(1,2))
plot(u133VsExon, cex = 0.5, pch = 4, col = "tomato", main = "P-values",
     xlab = "P   (U133)", ylab = "P   (Exon)")
plot(uhat, cex = 0.5, pch = 4, col = "tomato", main = "Ranked P-values",
     xlab = "rank(1-P)   (U133)", ylab = "rank(1-P)   (Exon)")

## End(Not run)

# Fitting using BFGS
fit <- fit.meta.GMCM(uhat, init.par = c(0.5, 1, 1, 0.5), pgtol = 1e-2,
                     method = "L-BFGS", positive.rho = TRUE, verbose = TRUE)

# Compute IDR values and classify
```

```
idr <- get.IDR(uhat, par = fit)
table(idr$K) # 1 = irreproducible, 2 = reproducible

## Not run:
# See clustering results
par(mfrow = c(1,2))
plot(u133VsExon, cex = 0.5, pch = 4, main = "Classified genes",
     col = c("tomato", "steelblue")[idr$K],
     xlab = "P-value (U133)", ylab = "P-value (Exon)")
plot(uhat, cex = 0.5, pch = 4, main = "Classified genes",
     col = c("tomato", "steelblue")[idr$K],
     xlab = "rank(1-P) (U133)", ylab = "rank(1-P) (Exon)")

## End(Not run)
```

---

as.theta *Coerce a list to a theta object*

Description

A function that attempts to coerce a theta-like list into a proper formatted object of class `theta`.

Usage

```
as.theta(x)
```

Arguments

`x`	A theta-like object that can be coerced.

Details

First, if the list is of length 3 and not 5, the number of components and dimension is assumed to be missing and added. Secondly, the class is added. Thirdly, names are added if needed. Next, matrix means and array covariances are coerced to list form. Covariances on array form are assumed to be `d` by `d` by `m`. Means on matrix form are as assumed to be `d` by `m`. I.e. rows correspond to the dimensions and columns to components, or the mean vectors as column vectors. Finally, the sum constraint of 1 for the mixture proportions is enforced.

Value

A theta object. See `rtheta`.

Examples

```
m <- 2
d <- 3
x <- list(m = m,
          d = d,
          pie = c(0.5, 0.5),
          mu = list(comp1=rep(0,d), comp2=rep(1,d)),
          sigma = list(comp1=diag(d), comp2=diag(d)))
print(x)
theta <- as.theta(x)
print(theta)

x2 <- unname(list( # Unnamed
  # missing m and d
  pie = c(1, 1),   # Does not sum to 1
  mu = simplify2array(list(comp1=rep(0,d), comp2=rep(1,d))), # matrix, not a list
  sigma = simplify2array(list(comp1=diag(d), comp2=diag(d)))  # array, not a list
))
theta2 <- as.theta(x2)
print(theta2)
```

`choose.theta` *Heuristically chosen starting value of theta*

Description

This function uses a k-means algorithm to heuristically select suitable starting values for the general model.

Usage

```
choose.theta(u, m, no.scaling = FALSE, ...)
```

Arguments

`u`	A matrix of (estimates of) realizations from the GMCM.
`m`	The number of components to be fitted.
`no.scaling`	Logical. If TRUE, no scaling of the means and variance-covariance matrices is done.
`...`	Arguments passed to `kmeans`.

Details

The function selects the centers from the k-means algorithm as an initial estimate of the means. The proportional sizes of the clusters are selected as the initial values of the mixture proportions. The within cluster standard deviations are squared and used as the variance of the clusters within each dimension. The correlations between each dimension are taken to be zero.

Value

A list of parameters for the GMCM model on the form described in rtheta.

Note

The function uses the kmeans function from the stats-package.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

Examples

```
set.seed(2)

# Simulating data
data1 <- SimulateGMCMData(n = 10000, m = 3, d = 2)
obs.data <- Uhat(data1$u)  # The ranked observed data

# Using choose.theta to get starting estimates
theta <- choose.theta(u = obs.data, m = 3)
print(theta)

# To illustrate theta, we can simulate from the model
data2 <- SimulateGMMData(n = 10000, theta = theta)

cols <- apply(get.prob(obs.data,theta),1,which.max)

# Plotting
par(mfrow = c(1,3))
plot(data1$z, main = "True latent GMM")
plot(Uhat(data1$u), col = cols,
     main = "Observed GMCM\nColoured by k-means clustering")
plot(data2$z, main = "initial GMM")

# Alteratively, theta can simply be plotted to illustrate the GMM density
par(mfrow = c(1,1))
plot(theta, add.ellipses = TRUE)
points(data2$z, pch = 16, cex = 0.4)
```

---

classify *Classify observations*

Description

Classify observations according to the maximum a posterior probabilites.

Usage

```
classify(x, theta)
```

Arguments

x	Either a matrix of A) observations where rows corresponds to obsercations and columns to dimensions or B) class probabilities where rows correspond to obsevations and columns to components.
theta	A list of parameters for the full model as described in rtheta. If theta is supplied, x are assumed to be observations (A). If theta is missing, x are assumed to be probabilites (B).

Value

A integer vector of class numbers with length equal to the number of rows in x.

See Also

get.prob

Examples

```
# Classify using probabilites (usually returned from get.prob)
probs <- matrix(runif(75), 25, 3)
classify(probs)

# Classify using a matrix of observations and theta
theta <- rtheta(d = 4, m = 3)
u <- SimulateGMCMData(n = 20, theta = theta)$u
classify(x = u, theta = theta)
```

---

dmvnormal — *Multivariate Gaussian density and simulation*

Description

Fast simulation from and evaluation of multivariate Gaussian probability densities.

Usage

```
dmvnormal(x, mu, sigma)

rmvnormal(n, mu, sigma)
```

Arguments

x	A p times k matrix of quantiles. Each rows correspond to a realization from the density and each column corresponds to a dimension.
mu	The mean vector of dimension k.
sigma	The variance-covariance matrix of dimension k times k.
n	The number of observations to be simulated.

Details

dmvnormal functions similarly to dmvnorm from the mvtnorm-package and likewise for rmvnormal and rmvnorm.

Value

dmvnormal returns a 1 by p matrix of the probability densities corresponding to each row of x. sigma. Each row corresponds to an observation.

rmvnormal returns a p by k matrix of observations from a multivariate normal distribution with the given mean mu and covariance

Author(s)

Anders Ellern Bilgrau

See Also

dmvnorm and rmvnorm in the mvtnorm-package.

Examples

```
dmvnormal(x = matrix(rnorm(300), 100, 3),
          mu = 1:3,
          sigma = diag(3))
rmvnormal(n = 10, mu = 1:4, sigma = diag(4))
```

EMAlgorithm — *EM algorithm for Gaussian mixture models*

Description

The regular expectation-maximization algorithm for general multivariate Gaussian mixture models.

Usage

```
EMAlgorithm(x, theta, m, eps = 1e-06, max.ite = 1e+05,
  trace.theta = FALSE, verbose = FALSE)
```

Arguments

x	A matrix of observations where each row correspond to an observation and each columns to a feature/variable.
theta	A list of parameters of class theta as described in rtheta. Optional. If not provided m should be given.
m	numeric. The number of components if theta is not supplied.
eps	The maximal required difference in successive likelihoods to establish convergence.

max.ite	The maximum number of iterations.
trace.theta	Logical. If TRUE, all estimates are stored and returned. Default is FALSE.
verbose	Set to TRUE for verbose output. Default is FALSE.

Details

Though not as versatile, the algorithm can be a faster alternative to Mclust in the mclust-package. If theta is not given, a k-means clustering is used to determine the initial theta.

Value

A list of length 3 with elements:

theta	A list of the estimated parameters as described in rtheta.
loglik.tr	A numeric vector of the log-likelihood trace.
kappa	A matrix where kappa[i,j] is the probability that x[i,] is realized from the j'th component.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

See Also

rtheta, PseudoEMAlgorithm

Examples

```
set.seed(3)
true.theta <- rtheta(d = 2, m = 3, method = "old")
true.theta$sigma <- lapply(true.theta$sigma, cov2cor) # Scale
## Not run:
plot(true.theta, nlevels = 20, add.ellipses = TRUE)

## End(Not run)

data <- SimulateGMCMData(n = 1000, theta = true.theta)
start.theta <- rtheta(d = 2, m = 3)
start.theta$mu <- t(kmeans(data$z, 3)$centers) # More sensible location estimates
start.theta <- as.theta(start.theta) # Coerce the matrix to a list
res <- GMCM:::EMAlgorithm(data$z, theta = start.theta)

par(mfrow = c(1,2))
plot(data$z, cex = 0.5, pch = 16, main = "Simulated data",
     col = rainbow(3)[data$K])
plot(data$z, cex = 0.5, pch = 16, main = "GMM clustering",
     col = rainbow(3)[apply(res$kappa,1,which.max)])
```

fit.full.GMCM *Estimate GMCM parameters of the general model*

Description

Estimates the parameters of general Gaussian mixture copula models (GMCM). The function finds the maximum likelihood estimate of a general GMCM with various optimization procedures. Note, all but the PEM methods provides the maximum likelihood estimate.

Usage

```
fit.full.GMCM(u, m, theta = choose.theta(u, m), method = c("NM",
  "SANN", "L-BFGS", "L-BFGS-B", "PEM"), max.ite = 1000, verbose = TRUE,
  ...)

fit.general.GMCM(u, m, theta = choose.theta(u, m), method = c("NM",
  "SANN", "L-BFGS", "L-BFGS-B", "PEM"), max.ite = 1000, verbose = TRUE,
  ...)
```

Arguments

`u`	An `n` by `d` matrix of marginally uniform observations. Rows corresponds to observations and columns to the dimensions of the variables. I.e. these are often ranked and scaled test statistics or other observations.
`m`	The number of components to be fitted.
`theta`	A list of parameters as defined in `rtheta`. If `theta` is not provided, then heuristic starting values are chosen using the k-means algorithm.
`method`	A character vector of length 1. The optimization method used. Should be either `"NM"`, `"SANN"`, `"L-BFGS"`, `"L-BFGS-B"`, or `"PEM"` which are the Nelder-Mead, Simulated Annealing, limited-memory quasi-Newton method, limited-memory quasi-Newton method with box constraints, and the pseudo EM algorithm, respectively. Default is `"NM"`. See `optim` for further details.
`max.ite`	The maximum number of iterations. If the `method` is `"SANN"` this is the number of iterations as there is no other stopping criterion. (See `optim`)
`verbose`	Logical. If `TRUE`, a trace of the parameter estimates is made.
`...`	Arguments passed to the `control`-list in `optim` when `method` is not equal to `"PEM"`. If `method` equals `"PEM"`, the arguments are passed to `PseudoEMAlgorithm` if the `method`.

Details

The `"L-BFGS-B"` method does not perform a transformation of the parameters and uses box constraints as implemented in `optim`.
Note that the many parameter configurations are poorly estimable or directly unidentifiable.

`fit.general.GMCM` is simply an alias of `fit.full.gmcm`.

Value

A list of parameters formatted as described in rtheta.

When method equals "PEM", a list of extra information (log-likelihood trace, the matrix of group probabilities, theta trace) is added as an attribute called "extra".

Note

All the optimization procedures are strongly dependent on the initial values and other parameters (such as the cooling scheme for method SANN). Therefore it is advisable to apply multiple different initial parameters (and optimization routines) and select the best fit.

The choose.theta itself chooses random a initialization. Hence, the output when theta is not directly supplied can vary.

See optim for further details.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Li, Q., Brown, J. B. J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. The Annals of Applied Statistics, 5(3), 1752-1779. doi:10.1214/11-AOAS466

Tewari, A., Giering, M. J., & Raghunathan, A. (2011). Parametric Characterization of Multimodal Distributions with Non-gaussian Modes. 2011 IEEE 11th International Conference on Data Mining Workshops, 286-292. doi:10.1109/ICDMW.2011.135

See Also

optim, get.prob

Examples

```
set.seed(17)
sim <- SimulateGMCMData(n = 1000, m = 3, d = 2)

# Plotting simulated data
par(mfrow = c(1,2))
plot(sim$z, col = rainbow(3)[sim$K], main = "Latent process")
plot(sim$u, col = rainbow(3)[sim$K], main = "GMCM process")

# Observed data
uhat <- Uhat(sim$u)

# The model should be fitted multiple times using different starting estimates
start.theta <- choose.theta(uhat, m = 3)  # Random starting estimate
res <- fit.full.GMCM(u = uhat, theta = start.theta,
                     method = "NM", max.ite = 3000,
                     reltol = 1e-2, trace = TRUE)  # Note, 1e-2 is too big
```

```
# Confusion matrix
Khat <- apply(get.prob(uhat, theta = res), 1, which.max)
table("Khat" = Khat, "K" = sim$K)  # Note, some components have been swapped

# Simulation from GMCM with the fitted parameters
simfit <- SimulateGMCMData(n = 1000, theta = res)

# As seen, the underlying latent process is hard to estimate.
# The clustering, however, is very good.
par(mfrow = c(2,2))
plot(simfit$z, col = simfit$K, main = "Model check 1\nSimulated GMM")
plot(simfit$u, col = simfit$K, main = "Model check 2\nSimulated GMCM")
plot(sim$u, col = Khat, main = "MAP clustering")
```

`fit.meta.GMCM` *Estimate GMCM parameters of the special model*

Description

This function estimates the parameters of the special restricted Gaussian mixture copula model (GMCM) proposed by Li et. al. (2011). It is used to perform reproducibility (or meta) analysis using GMCMs. It features various optimization routines to identify the maximum likelihood estimate of the special GMCMs.

Usage

```
fit.meta.GMCM(u, init.par, method = c("NM", "SANN", "L-BFGS", "L-BFGS-B",
  "PEM"), max.ite = 1000, verbose = TRUE, positive.rho = TRUE,
  trace.theta = FALSE, ...)

fit.special.GMCM(u, init.par, method = c("NM", "SANN", "L-BFGS",
  "L-BFGS-B", "PEM"), max.ite = 1000, verbose = TRUE,
  positive.rho = TRUE, trace.theta = FALSE, ...)
```

Arguments

`u`	An `n` by `d` matrix of test statistics. Rows correspond to features and columns to experiments. Larger values are assumed to be indicative of stronger evidence and reproducibility.
`init.par`	A 4-dimensional vector of the initial parameters where, `init.par[1]` is the mixture proportion of spurious signals, `init.par[2]` is the mean, `init.par[3]` is the standard deviation, `init.par[4]` is the correlation.
`method`	A character vector of length 1. The optimization method used. Should be either `"NM"`, `"SANN"`, `"L-BFGS"`, `"L-BFGS-B"`, or `"PEM"` which are abbreviations of Nelder-Mead, Simulated Annealing, limited-memory quasi-Newton method, limited-memory quasi-Newton method with box constraints, and the pseudo EM algorithm, respectively. Default is `"NM"`. See `optim` for further details.

max.ite	The maximum number of iterations. If the method is "SANN" this is the number of iterations as there is no other stopping criterion. (See optim)
verbose	Logical. If TRUE, the log-likelihood values are printed.
positive.rho	logical. If TRUE, the correlation parameter is restricted to be positive.
trace.theta	logical. Extra convergence information is appended as a list to the output returned if TRUE. The exact behavior is dependent on the value of method. If method equals "PEM", the argument is passed to trace.theta in PseudoEMAlgorithm. Otherwise it is passed to the control argument trace in optim.
...	Arguments passed to the control-list in optim or PseudoEMAlgorithm if method is "PEM".

Details

The "L-BFGS-B" method does not perform a transformation of the parameters.

fit.special.GMCM is simply an alias of fit.meta.gmcm.

Value

A vector par of length 4 of the fitted parameters where par[1] is the probability of being from the first (or null) component, par[2] is the mean, par[3] is the standard deviation, and par[4] is the correlation.

If trace.theta is TRUE, then a list is returned where the first entry is as described above and the second entry is the trace information (dependent of method.).

Note

Simulated annealing is strongly dependent on the initial values and the cooling scheme.

See optim for further details.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Li, Q., Brown, J. B. J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. The Annals of Applied Statistics, 5(3), 1752-1779. doi:10.1214/11-AOAS466

See Also

optim

Examples

```
set.seed(1)

# True parameters
true.par <- c(0.9, 2, 0.7, 0.6)
# Simulation of data from the GMCM model
data <- SimulateGMCMData(n = 1000, par = true.par)
uhat <- Uhat(data$u) # Ranked observed data

init.par <- c(0.5, 1, 0.5, 0.9)  # Initial parameters

# Optimization with Nelder-Mead
nm.par   <- fit.meta.GMCM(uhat, init.par = init.par, method = "NM")

## Not run:
# Comparison with other optimization methods
# Optimization with simulated annealing
sann.par <- fit.meta.GMCM(uhat, init.par = init.par, method = "SANN",
                          max.ite = 3000, temp = 1)
# Optimization with the Pseudo EM algorithm
pem.par  <- fit.meta.GMCM(uhat, init.par = init.par, method = "PEM")

# The estimates agree nicely
rbind("True" = true.par, "Start" = init.par,
      "NM" = nm.par, "SANN" = sann.par, "PEM" = pem.par)

## End(Not run)

# Get estimated cluster
Khat <- get.IDR(x = uhat, par = nm.par)$Khat
plot(uhat, col = Khat, main = "Clustering\nIDR < 0.05")
```

`freshVsFrozen` *Reproducibility between Fresh and Frozen B-cell subtypes*

Description

This dataset contains a `data.frame` of t-scores (from a Linear mixed effects model) and p-values for differential expression between pre (Im, N) and post germinal (M, PB) centre cells within peripheral blood. The first and second column contain the the test for the hypothesis of no differentially expression between pre and post germinal cells for the freshly sorted and gene profiled cells. The third and fourth column contain the the test for the hypothesis of no differentially expression between pre and post germinal cells for the cryopreserved (frozen), thawed, sorted, and gene profiled cells. The fifth and sixth column contain the the test for the hypothesis of no differentially expression between fresh and frozen cells. The used array type was Affymetrix Human Exon 1.0 ST microarray.

Format

The format of the `data.frame` is:

```
'data.frame': 18708 obs. of 6 variables:
$ PreVsPost.Fresh.tstat : num -1.073 -0.381 -1.105 -0.559 -1.054 ...
$ PreVsPost.Fresh.pval : num 0.283 0.703 0.269 0.576 0.292 ...
$ PreVsPost.Frozen.tstat: num -0.245 -0.731 -0.828 -0.568 -1.083 ...
$ PreVsPost.Frozen.pval : num 0.806 0.465 0.408 0.57 0.279 ...
$ FreshVsFrozen.tstat : num 0.836 1.135 -0.221 0.191 -0.783 ...
$ FreshVsFrozen.pval : num 0.403 0.256 0.825 0.849 0.434 ...
```

Details

Further details can be found in Rasmussen and Bilgrau et al. (2015).

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Rasmussen SM, Bilgrau AE, Schmitz A, Falgreen S, Bergkvist KS, Tramm AM, Baech J, Jacobsen CL, Gaihede M, Kjeldsen MK, Boedker JS, Dybkaer K, Boegsted M, Johnsen HE (2015). "Stable Phenotype Of B-Cell Subsets Following Cryopreservation and Thawing of Normal Human Lymphocytes Stored in a Tissue Biobank." Cytometry Part B: Clinical Cytometry, 88(1), 40-49.

Examples

```
data(freshVsFrozen)
str(freshVsFrozen)

# Plot P-values
plot(freshVsFrozen[,c(2,4)], cex = 0.5)

# Plot ranked and scaled P-values
plot(Uhat(abs(freshVsFrozen[,c(1,3)])), cex = 0.5)
```

`full2meta` *Convert between parameter formats*

Description

These functions converts/coerces the parameters between the general Gaussian mixture (copula) model and the special GMCM. Most functions of the GMCM packages use the `theta` format described in `rtheta`.

Usage

```
full2meta(theta)

meta2full(par, d)
```

Arguments

theta	A list of parameters for the full model. Formatted as described in rtheta.
par	A vector of length 4 where par[1] is the probability of coming from the first component, par[2] is the mean value, par[3] is the standard deviation, and par[4] is the correlation of the reproducible component.
d	An integer giving the dimension of the mixture distribution.

Details

If a theta is supplied which is not on the form of Li et. al. (2011) the output is coerced by simply picking the first element of the second component mean vector as mean, the square roof of the first diagonal entry of the second component covariance matrix as standard deviation, and first off-diagonal entry as correlation (properly scaled).

Value

full2meta returns a numeric vector of length 4 formatted as par.

meta2full returns a formatted 'theta' list of parameters as described by rtheta.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Li, Q., Brown, J. B. J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. The Annals of Applied Statistics, 5(3), 1752-1779. doi:10.1214/11-AOAS466

Tewari, A., Giering, M., & Raghunathan, A. (2011). Parametric Characterization of Multimodal Distributions with Non-gaussian Modes. IEEE 11th International Conference on Data Mining Workshops, 2011, 286-292. doi:10.1109/ICDMW.2011.135

See Also

rtheta

Examples

```
theta <- GMCM:::rtheta(m = 2, d = 2)
print(par <- full2meta(theta))
print(theta.special.case <- meta2full(par, d = 2))
```

`get.IDR` *Posterior class probabilities, local, and adjusted IDRs.*

Description

Functions for computing posterior cluster probabilities (`get.prob`) in the general GMCM as well as local and adjusted irreproducibility discovery rates (`get.IDR`) in the special GMCM.

Usage

```
get.IDR(x, par, threshold = 0.05, ...)

get.prob(x, theta, ...)
```

Arguments

`x`	A `matrix` of observations where rows corresponds to features and columns to studies.
`par`	A vector of length 4 where `par[1]` is mixture proportion of the irreproducible component, `par[2]` is the mean value, `par[3]` is the standard deviation, and `par[4]` is the correlation of the reproducible component.
`threshold`	The threshold level of the IDR rate.
`...`	Arguments passed to `qgmm.marginal`.
`theta`	A list of parameters for the full model as described in `rtheta`.

Value

`get.IDR` returns a list of length 5 with elements:

`idr`	A vector of the local idr values. I.e. the posterior probability that `x[i, ]` belongs to the irreproducible component.
`IDR`	A vector of the adjusted IDR values.
`l`	The number of reproducible features at the specified `threshold`.
`threshold`	The IDR threshold at which features are deemed reproducible.
`Khat`	A vector signifying whether the corresponding feature is reproducible or not.

`get.prob` returns a matrix where entry `(i,j)` is the posterior probability that the observation `x[i, ]` belongs to cluster `j`.

Note

From **GMCM** version 1.1 `get.IDR` has been an internal function. Use `get.prop` or `get.IDR` instead. The function can still be accessed with `GMCM:::get.idr`. `get.idr` returns a vector where the i'th entry is the posterior probability that observation i is irreproducible. It is a simple wrapper for `get.prob`.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Li, Q., Brown, J. B. J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. The Annals of Applied Statistics, 5(3), 1752-1779. doi:10.1214/11-AOAS466

Tewari, A., Giering, M., & Raghunathan, A. (2011). Parametric Characterization of Multimodal Distributions with Non-gaussian Modes. IEEE 11th International Conference on Data Mining Workshops, 2011, 286-292. doi:10.1109/ICDMW.2011.135

Examples

```
set.seed(1123)

# True parameters
true.par <- c(0.9, 2, 0.7, 0.6)

# Simulation of data from the GMCM model
data <-  SimulateGMCMData(n = 1000, par = true.par, d = 2)

# Initial parameters
init.par <- c(0.5, 1, 0.5, 0.9)

# Nelder-Mead optimization
nm.par   <- fit.meta.GMCM(data$u, init.par = init.par, method = "NM")

# Get IDR values
res <- get.IDR(data$u, nm.par, threshold = 0.05)

# Plot results
plot(data$u, col = res$Khat, pch = c(3,16)[data$K])
```

---

goodness.of.fit *Goodness of fit for the general GMCM*

Description

Compute goodness of fit as described in AIC. The number of parameters used correspond to the number of variables free to vary in the general model.

Usage

```
goodness.of.fit(theta, u, method = c("AIC", "BIC"), k = 2)
```

Arguments

theta	A list of parameters as defined in rtheta. For t this function, it will usually be the output of fit.full.GMCM.
u	An n by d matrix of marginally uniform observations. Rows corresponds to observations and columns to the dimensions of the variables. I.e. these are often ranked and scaled test statistics or other observations.
method	A character of length 1 which specifies the goodness of fit to compute. Default is "AIC". "BIC" is also a option.
k	A integer specifying the default used constant "k" in AIC. See AIC.

Value

A single number giving the goodness of fit as requested.

Examples

```
set.seed(2)
data(u133VsExon)
u <- Uhat(u133VsExon[sample(19577, 500), ])  # Subset for faster fitting
theta1 <- fit.full.GMCM(u, m = 2, method = "L-BFGS")
goodness.of.fit(theta1, u)  # AIC
goodness.of.fit(theta1, u, method = "BIC")
## Not run:
theta2 <- fit.full.GMCM(u, m = 3, method = "L-BFGS")
goodness.of.fit(theta2, u)
goodness.of.fit(theta2, u, method = "BIC")

## End(Not run)
```

---

is.theta — *Check if parameters are valid*

Description

Function to check whether the argument is coherent and in the correct format.

Usage

```
is.theta(theta, check.class = TRUE)
```

Arguments

theta	A list on the theta-form described in rtheta
check.class	Logical. If TRUE, the class of theta is also checked.

Value

logical. Returns TRUE if theta is coherent and in the correct format. Otherwise, the function returns FALSE with an accompanying warning message of the problem.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

See Also

rtheta

Examples

```
theta1 <- rtheta()  # Create a random correctly formatted theta
is.theta(theta1)

theta2 <- rtheta(d = 3, m = 5)
theta2$m <- 6  # m is now incoherent with the number of components
is.theta(theta2)

theta3 <- rtheta(d = 4, m = 2)
theta3$sigma$comp1[1, 2] <- 0  # Making the covariance matrix non-symmetric
is.theta(theta3)

theta4 <- rtheta(d = 10, m = 10)
theta4$sigma$comp1[1, 1] <- 0  # Destroy positive semi-definiteness
is.theta(theta4)

theta5 <- rtheta()
names(theta5) <- c("m", "d", "prop", "mu", "sigmas") # Incorrect names
is.theta(theta5)
```

---

plot.theta *Plotting method for "theta" objects*

Description

Visualizes the chosen dimensions of the theta object graphically by the GMM density and possibly the individual gaussian components.

Usage

```
## S3 method for class 'theta'
plot(x, which.dims = c(1L, 2L), n.sd = qnorm(0.99),
  add.means = TRUE, ..., add.ellipses = FALSE)
```

Arguments

`x`	An object of class `theta`.
`which.dims`	An integer vector of length 2 choosing which two dimensions to plot.
`n.sd`	An integer choosing the number of standard deviations in each dimension to determine the plotting window.
`add.means`	logical. If TRUE, dots corresponding to the means are added to the plot.
`...`	Arguments passed to `contour`.
`add.ellipses`	logical. If TRUE, ellipses outlining a 95% confidence regions for each component are added in the bivariate multivariate distribution defined by theta and `which.dims`.

Value

Plots via the `contour` function. Invisibly returns a list with x, y, z coordinates that is passed to contour.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

Examples

```
set.seed(5)
theta <- rtheta(d = 3, m = 2)
## Not run:
plot(theta)
plot(theta, col = "blue", asp = 1, add.means = FALSE)
plot(theta, col = "blue", asp = 1, add.means = TRUE)
plot(theta, which.dims = c(3L, 2L), asp = 1)

## End(Not run)
plot(theta, asp = 1, n.sd = 3, add.ellipses = TRUE,
     nlevels = 40, axes = FALSE,
     xlab = "Dimension 1", ylab = "Dimension 2")
```

`print.theta` *Print method for theta class*

Description

Print method for theta class

Usage

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

Arguments

`x`	A `theta` object. See `rtheta`.
`...`	Arguments to be passed to subsequent methods.

Value

Invisibly returns `x`.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

Examples

```
theta <- rtheta()
print(theta)
```

`rtheta` *Get random parameters for the Gaussian mixture (copula) model*

Description

Generate a random set parameters for the Gaussian mixture model (GMM) and Gaussian mixture copula model (GMCM). Primarily, it provides an easy prototype of the `theta`-format used in **GMCM**.

Usage

```
rtheta(m = 3, d = 2, method = c("old", "EqualSpherical",
  "UnequalSpherical", "EqualEllipsoidal", "UnequalEllipsoidal"))
```

Arguments

`m`	The number of components in the mixture.
`d`	The dimension of the mixture distribution.
`method`	The method by which the theta should be generated. See details. Defaults to `"old"` which is the regular "old" behavior.

Details

Depending on the `method` argument the parameters are generated as follows. The new behavior is inspired by the simulation scenarios in Friedman (1989) but not exactly the same.

- `pie` is generated by m draws of a chi-squared distribution with $3m$ degrees of freedom divided by their sum. If `method = "old"` the uniform distribution is used instead.
- `mu` is generated by m i.i.d. d-dimensional zero-mean normal vectors with covariance matrix `100I`. (unchanged from the old behavior)

- sigma is dependent on method. The covariance matrices for each component are generated as follows. If the method is
  - "EqualSpherical", then the covariance matrices are the identity matrix and thus are all equal and spherical.
  - "UnequalSpherical", then the covariance matrices are scaled identity matrices. In component h, the covariance matrix is hI
  - "EqualEllipsoidal", then highly elliptical covariance matrices which equal for all components are used. The square root of the d eigenvalues are chosen equidistantly on the interval 10 to 1 and a randomly (uniformly) oriented orthonormal basis is chosen and used for all components.
  - "UnqualEllipsoidal", then highly elliptical covariance matrices different for all components are used. The eigenvalues of the covariance matrices equal as in all components as in "EqualEllipsoidal". However, they are all randomly (uniformly) oriented (unlike as described in Friedman (1989)).
  - "old", then the old behavior is used. The old behavior differs from "EqualEllipsoidal" by using the absolute value of d zero-mean i.i.d. normal eigenvalues with a standard deviation of 8.

  In all cases, the orientation is selected uniformly.

Value

A named list of parameters with the 4 elements:

m	An integer giving the number of components in the mixture. Default is 3.
d	An integer giving the dimension of the mixture distribution. Default is 2.
pie	A numeric vector of length m of mixture proportions between 0 and 1 which sums to one.
mu	A list of length m of numeric vectors of length d for each component.
sigma	A list of length m of variance-covariance matrices (of size d times d) for each component.

Note

The function is.theta checks whether or not theta is in the correct format.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Friedman, Jerome H. "Regularized discriminant analysis." Journal of the American statistical association 84.405 (1989): 165-175.

See Also

is.theta

Examples

```
rtheta()

rtheta(d = 5, m = 2)

rtheta(d = 3, m = 2, method = "EqualEllipsoidal")

test <- rtheta()
is.theta(test)

summary(test)
print(test)
plot(test)

## Not run:
A <- SimulateGMMData(n = 100, rtheta(d = 2, method = "EqualSpherical"))
plot(A$z, col = A$K, pch = A$K, asp = 1)
B <- SimulateGMMData(n = 100, rtheta(d = 2, method = "UnequalSpherical"))
plot(B$z, col = B$K, pch = B$K, asp = 1)
C <- SimulateGMMData(n = 100, rtheta(d = 2, method = "EqualEllipsoidal"))
plot(C$z, col = C$K, pch = C$K, asp = 1)
D <- SimulateGMMData(n = 100, rtheta(d = 2, method = "UnequalEllipsoidal"))
plot(D$z, col = D$K, pch = D$K, asp = 1)
## End(Not run)
```

`runGMCM` *Run the GMCM shiny application*

Description

Function for starting a local instance of the GMCM shiny application. The online application is found at `https://gmcm.shinyapps.io/GMCM/`.

Usage

```
runGMCM(...)
```

Arguments

`...`	Arguments passed to `runApp`.

Value

Retuns nothing (usually). See `runApp`. Exit or stop the app by interrupting R.

See Also

`runApp`

Examples

```
## Not run:
runGMCM()
runGMCM(launch.browser = FALSE, port = 1111)
# Open browser and enter URL http://127.0.0.1:1111/

## End(Not run)
```

SimulateGMCMData *Simulation from Gaussian mixture (copula) models*

Description

Easy and fast simulation of data from Gaussian mixture copula models (GMCM) and Gaussian mixture models (GMM).

Usage

```
SimulateGMCMData(n = 1000, par, d = 2, theta, ...)

SimulateGMMData(n = 1000, theta = rtheta(...), ...)
```

Arguments

`n`	A single integer giving the number of realizations (observations) drawn from the model. Default is 1000.
`par`	A vector of parameters of length 4 where `par[1]` is the mixture proportion, `par[2]` is the mean, `par[3]` is the standard deviation, and `par[4]` is the correlation.
`d`	The number of dimensions (or, equivalently, experiments) in the mixture distribution.
`theta`	A list of parameters for the model as described in `rtheta`.
`...`	In `SimulateGMCMData` the arguments are passed to `SimulateGMMData`. In `SimulateGMMData` the arguments are passed to `rtheta`.

Details

The functions provide simulation of n observations and d-dimensional GMCMs and GMMs with provided parameters. The `par` argument specifies the parameters of the Li et. al. (2011) GMCM. The `theta` argument specifies an arbitrary GMCM of Tewari et. al. (2011). Either one can be supplied. If both are missing, random parameters are chosen for the general model.

Value

`SimulateGMCMData` returns a list of length 4 with elements:

- `u` A matrix of the realized values of the GMCM.
- `z` A matrix of the latent GMM realizations.
- `K` An integer vector denoting the component from which the realization comes.
- `theta` A list containing the used parameters for the simulations with the format described in `rtheta`.

`SimulateGMMData` returns a list of length 3 with elements:

- `z` A matrix of GMM realizations.
- `K` An integer vector denoting the component from which the realization comes.
- `theta` As above and in `rtheta`.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Li, Q., Brown, J. B. J. B., Huang, H., & Bickel, P. J. (2011). Measuring reproducibility of high-throughput experiments. The Annals of Applied Statistics, 5(3), 1752-1779. doi:10.1214/11-AOAS466

Tewari, A., Giering, M. J., & Raghunathan, A. (2011). Parametric Characterization of Multimodal Distributions with Non-gaussian Modes. 2011 IEEE 11th International Conference on Data Mining Workshops, 286-292. doi:10.1109/ICDMW.2011.135

See Also

`rtheta`

Examples

```
set.seed(2)

# Simulation from the GMM
gmm.data1 <- SimulateGMMData(n = 200, m = 3, d = 2)
str(gmm.data1)

# Plotting the simulated data
plot(gmm.data1$z, col = gmm.data1$K)

# Simulation from the GMCM
gmcm.data1 <- SimulateGMCMData(n = 1000, m = 4, d = 2)
str(gmcm.data1)

# Plotthe 2nd simulation
par(mfrow = c(1,2))
```

```
plot(gmcm.data1$z, col = gmcm.data1$K)
plot(gmcm.data1$u, col = gmcm.data1$K)

# Simulation from the special case of GMCM
theta <- meta2full(c(0.7, 2, 1, 0.7), d = 3)
gmcm.data2 <- SimulateGMCMData(n = 5000, theta = theta)
str(gmcm.data2)

# Plotting the 3rd simulation
par(mfrow=c(1,2))
plot(gmcm.data2$z, col = gmcm.data2$K)
plot(gmcm.data2$u, col = gmcm.data2$K)
```

`summary.theta` *Summary method for theta class*

Description

Summary method for theta class

Usage

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

Arguments

`object`	A `theta` object. See `rtheta`.
`...`	arguments to be passed to subsequent methods

Value

Invisibly returns x.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

Examples

```
theta <- rtheta()
summary(theta)
```

`u133VsExon` *Reproducibility between U133 plus 2 and Exon microarrays*

Description

This dataset contains a `data.frame` of unadjusted P-values for differential expression between germinal center cells and other B-cells within tonsils for two different experiments. The experiments differ primarily in the microarray platform used. The first column corresponds the evidence from the Affymetrix GeneChip Human Genome U133 Plus 2.0 Array. The second column corresponds to the Affymetrix GeneChip Human Exon 1.0 ST Array.

Format

The format of the `data.frame` is:

`'data.frame': 19577 obs. of 2 variables:`
`$ u133: num 0.17561 0.00178 0.005371 0.000669 0.655261 ...`
`$ exon: num 1.07e-01 6.74e-10 1.51e-03 6.76e-05 3.36e-01 ...`

Details

Further details can be found in Bergkvist et al. (2014) and Rasmussen and Bilgrau et al. (2014).

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

References

Bergkvist, Kim Steve, Mette Nyegaard, Martin Boegsted, Alexander Schmitz, Julie Stoeve Boedker, Simon Mylius Rasmussen, Martin Perez-Andres et al. (2014). "Validation and Implementation of a Method for Microarray Gene Expression Profiling of Minor B-Cell Subpopulations in Man". BMC immunology, 15(1), 3.

Rasmussen SM, Bilgrau AE, Schmitz A, Falgreen S, Bergkvist KS, Tramm AM, Baech J, Jacobsen CL, Gaihede M, Kjeldsen MK, Boedker JS, Dybkaer K, Boegsted M, Johnsen HE (2015). "Stable Phenotype Of B-Cell Subsets Following Cryopreservation and Thawing of Normal Human Lymphocytes Stored in a Tissue Biobank." Cytometry Part B: Clinical Cytometry, 88(1), 40-49.

Examples

```
data(u133VsExon)
str(u133VsExon)

# Plot P-values
plot(u133VsExon, cex = 0.5)

# Plot ranked and scaled P-values
plot(Uhat(1-u133VsExon), cex = 0.5)
```

`Uhat` *Fast ranking function*

Description

Function for computing the scaled ranks for each column of the input matrix. In other words, the values are ranked column-wise and divided by `nrow(x) + 1`. A "1334" ranking scheme is used where the lowest values is awarded rank 1, second lowest value rank 2, and ties are given the maximum available rank.

Usage

```
Uhat(x)
```

Arguments

x	A `numeric` `matrix` of observations to be ranked. Rows correspond to features and columns to experiments.

Value

A `matrix` with the same dimensions as `x` of the scaled ranks.

Author(s)

Anders Ellern Bilgrau <anders.ellern.bilgrau@gmail.com>

See Also

`SimulateGMMData`, `SimulateGMCMData`

Examples

```
data <- SimulateGMMData()
par(mfrow = c(1,2))
plot(data$z, xlab = expression(z[1]), ylab = expression(z[2]))
plot(Uhat(data$z),
     xlab = expression(hat(u)[1]),
     ylab = expression(hat(u)[2]))
```

Index