Package ‘NPflow’

January 13, 2024

Type Package

Title Bayesian Nonparametrics for Automatic Gating of Flow-Cytometry Data

Version 0.13.5

Date 2024-01-13

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.1), Rcpp (>= 0.12.11), truncnorm

Imports graphics, stats, grDevices, ellipse, fastcluster, ggplot2, pheatmap, reshape2, GGally

Suggests foreach, parallel, doParallel, itertools, microbenchmark


License LGPL-3 | file LICENSE

BugReports https://github.com/sistm/NPflow/issues

Encoding UTF-8

RoxygenNote 7.3.0

NeedsCompilation yes

Author Boris P Hejblum [aut, cre], Chariff Alkhassim [aut], Francois Caron [aut]

Maintainer Boris P Hejblum <boris.hejblum@u-bordeaux.fr>

Repository CRAN

Date/Publication 2024-01-13 10:00:02 UTC

R topics documented:

NPflow-package ................................................................. 3
burn.DPMMclust ................................................................. 4
R topics documented:

- cluster_est_binder ........................................... 4
- cluster_est_Fmeasure ........................................ 5
- cluster_est_Mbinder_norm .................................. 6
- cluster_est_pear ............................................. 7
- cytoScatter .................................................. 8
- DPMGibbsN .................................................... 10
- DPMGibbsN_parallel ......................................... 14
- DPMGibbsN_SeqPrior ......................................... 17
- DPMGibbsSkewN ............................................... 21
- DPMGibbsSkewN_parallel .................................... 29
- DPMGibbsSkewT ............................................... 32
- DPMGibbsSkewT_parallel .................................... 37
- DPMGibbsSkewT_SeqPrior .................................... 41
- DPMGibbsSkewT_SeqPrior_parallel ......................... 44
- DPMpost ....................................................... 48
- evalClustLoss ................................................ 53
- Flimited ....................................................... 54
- FmeasureC ..................................................... 55
- FmeasureC_no0 ............................................... 55
- Fmeasure_costC ............................................... 56
- lgamma_mv ..................................................... 57
- MAP_sNiW_mmEM ............................................. 57
- MLE_gamma ..................................................... 60
- MLE_NiW_mmEM .............................................. 61
- MLE_sNiW ..................................................... 62
- MLE_sNiW_mmEM ............................................. 63
- mmNiWpdf ..................................................... 65
- mmNiWpdfC .................................................... 66
- mmsNiWlogpdf ................................................. 67
- mmsNiWpdfC ................................................... 68
- mmvnpdfC ..................................................... 69
- mmvsnpdfC .................................................... 70
- mmvstpdfC ..................................................... 72
- mmvtpdfC ..................................................... 73
- mvnlikC ....................................................... 75
- mvnpdf ......................................................... 76
- mvnpdfC ....................................................... 77
- mvsnlikC ...................................................... 78
- mvsnpdf ....................................................... 79
- mvstlikC ....................................................... 80
- mvstpdf ....................................................... 81
- mvtpdf ......................................................... 82
- NuMatParC ..................................................... 83
- plot_ConvDPM ............................................... 83
- plot_DPM ....................................................... 84
- plot_DPMsn ................................................... 85
- plot_DPMst ................................................... 86
- postProcess.DPMMclust ....................................... 87
Description

Dirichlet process mixture of multivariate normal, skew normal or skew t-distributions modeling oriented towards flow-cytometry data pre-processing applications.

Details

Package: NPflow
Type: Package
Version: 0.13.5
Date: 2024-01-13
License: LGPL-3

The main function in this package is \texttt{DPMpost}.

Author(s)

Boris P. Hejblum, Chariff Alkhassim, Francois Caron — Maintainer: Boris P. Hejblum

References


See Also

Useful links:
- Report bugs at https://github.com/sistm/NPflow/issues
burn.DPMMclust  

Burning MCMC iterations from a Dirichlet Process Mixture Model.

Description

Utility function for burning MCMC iteration from a DPMMclust object.

Usage

burn.DPMMclust(x, burnin = 0, thin = 1)

Arguments

- **x**  
a DPMMclust object.
- **burnin**  
the number of MCMC iterations to burn (default is 0).
- **thin**  
the spacing at which MCMC iterations are kept. Default is 1, i.e. no thinning.

Value

a DPMMclust object minus the burnt iterations

Author(s)

Boris Hejblum

See Also

summary.DPMMclust

cluster_est_binder  

Point estimate of the partition for the Binder loss function

Description

Get a point estimate of the partition using the Binder loss function.

Usage

cluster_est_binder(c, logposterior)

Arguments

- **c**  
a list of vector of length n. c[[j]][i] is the cluster allocation of observation i=1,...,n at iteration j=1,...,N.
- **logposterior**  
vector of logposterior corresponding to each partition from c used to break ties when minimizing the cost function
Value

a list:

c_est: a vector of length n. Point estimate of the partition
cost: a vector of length N. cost[j] is the cost associated to partition c[[j]]
similarity: matrix of size n x n. Similarity matrix (see similarityMat)

opt_ind: the index of the optimal partition among the MCMC iterations.

Author(s)

Francois Caron, Boris Hejblum

References


DB Dahl, Model-Based Clustering for Expression Data via a Dirichlet Process Mixture Model, Bayesian Inference for Gene Expression and Proteomics, K-A Do, P Muller, M Vannucci (Eds.), Cambridge University Press, 2006.

See Also

similarityMat similarityMatC

cluster_est_Fmeasure  Point estimate of the partition using the F-measure as the cost function.

Description

Get a point estimate of the partition using the F-measure as the cost function.

Usage

cluster_est_Fmeasure(c, logposterior)

Arguments

c a list of vector of length n. c[[j]][i] is the cluster allocation of observation i=1...n at iteration j=1...N.

logposterior a vector of logposterior corresponding to each partition from c used to break ties when minimizing the cost function
Value

- c_est: a vector of length n. Point estimate of the partition
- cost: a vector of length N. cost[j] is the cost associated to partition c[[j]]
- similarity: matrix of size n x n. Similarity matrix (see similarityMat)
- opt_ind: the index of the optimal partition among the MCMC iterations.

Author(s)

Francois Caron, Boris Hejblum

See Also

similarityMat

---

cluster_est_Mbinder_norm

Point estimate of the partition using a modified Binder loss function

Description

Get a point estimate of the partition using a modified Binder loss function for Gaussian components

Usage

cluster_est_Mbinder_norm(c, Mu, Sigma, lambda = 0, a = 1, b = a, logposterior)

Arguments

- c: a list of vector of length n. c[[j]][i] is the cluster allocation of observation i=1...n at iteration j=1...N.
- Mu: is a list of length n composed of p x 1 matrices. Where l is the maximum number of components per partition.
- Sigma: is list of length n composed of arrays containing a maximum of l p x p covariance matrices.
- lambda: is a nonnegative tuning parameter allowing further control over the distance function. Default is 0.
- a: nonnegative constant seen as the unit cost for pairwise misclassification. Default is 1.
- b: nonnegative constant seen as the unit cost for the other kind of pairwise misclassification. Default is 1.
- logposterior: vector of logposterior corresponding to each partition from c used to break ties when minimizing the cost function
cluster_est_pear

Details

Note that the current implementation only allows Gaussian components.

The modified Binder loss function takes into account the distance between mixture components using the Bhattacharyya distance.

Value

alist:

c_est: a vector of length n. Point estimate of the partition

cost: a vector of length N. cost[j] is the cost associated to partition c[[j]]
similarity: matrix of size n x n. Similarity matrix (see similarityMat)
opt_ind: the index of the optimal partition among the MCMC iterations.

Author(s)

Chariff Alkhassim

References


See Also

similarityMat similarityMatC similarityMat_nocostC

cluster_est_pear

Gets a point estimate of the partition using posterior expected adjusted Rand index (PEAR)

Description

Gets a point estimate of the partition using posterior expected adjusted Rand index (PEAR)

Usage

cluster_est_pear(c)

Arguments

c a list of vector of length n. c[[j]][i] is the cluster allocation of observation i=1...n at iteration j=1...N.
Value

alist:
c_est: a vector of length n. Point estimate of the partition
pear: a vector of length N. pear[j] is the posterior expected adjusted Rand index associated to partition c[[j]]
similarity: matrix of size n x n. Similarity matrix (see similarityMat)
opt_ind: the index of the optimal partition among the MCMC iterations.

Author(s)

Chariff Alkhassim

References


See Also

similarityMat similarityMatC

cytoScatter Scatterplot of flow cytometry data

Description

Scatterplot of flow cytometry data

Usage

cytoScatter(
  cytomatrix,
  dims2plot = c(1, 2),
  gating = NULL,
  scale_log = FALSE,
  xlim = NULL,
  ylim = NULL,
  gg.add = list(theme())
)
cytoScatter

Arguments

cytomatrix a p x n data matrix, of n cell observations measured over p markers.
dims2plot a vector of length at least 2, indicating of the dimensions to be plotted. Default is c(1, 2).
gating an optional vector of length n indicating a known gating of the cells to be displayed. Default is NULL in which case no gating is displayed.
scale_log a logical Flag indicating whether the data should be plotted on the log scale. Default is FALSE.
xlim a vector of length 2 to specify the x-axis limits. Only used if dims2plot is of length 2. Default is the data range.
ylim a vector of length 2 to specify the y-axis limits. Only used if dims2plot is of length 2. Default is the data range.

gg.add A list of instructions to add to the ggplot2 instruction (see gg-add). Default is list(theme()), which adds nothing to the plot.

Examples

```
rm(list=ls())
#Number of data
n <- 500
#n <- 2000
set.seed(1234)
#set.seed(123)
#set.seed(4321)

# Sample data
m <- matrix(nrow=2, ncol=4, c(-1, 1, 1.5, 2, 2, -2, -1.5, -2))
p <- c(0.2, 0.1, 0.4, 0.3) # frequence des clusters
sdev <- array(dim=c(2,2,4))
sdev[, ,1] <- matrix(nrow=2, ncol=2, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=2, ncol=2, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=2, ncol=2, c(0.3, 0.15, 0.15, 0.3))
sdev[, ,4] <- .3*diag(2)
c <- rep(0,n)
z <- matrix(0, nrow=2, ncol=n)
for(k in 1:n){
  c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
z[,k] <- m[, c[k]] + sdev[, , c[k]]%*%matrix(rnorm(2, mean = 0, sd = 1), nrow=2, ncol=1)
}
```
cyroScatter(z)
Slice Sampling of the Dirichlet Process Mixture Model with a prior on \( \alpha \)

Description

Slice Sampling of the Dirichlet Process Mixture Model with a prior on \( \alpha \)

Usage

\[
\text{DPMGibbsN}(z, \text{hyperG0, } a = 1e-04, b = 1e-04, N, \text{doPlot = TRUE, nbclust_init = 30, plotevery = N/10, diagVar = TRUE, use_variance_hyperprior = TRUE, verbose = TRUE, ...})
\]

Arguments

- \( z \)
  data matrix \( d \times n \) with \( d \) dimensions in rows and \( n \) observations in columns.
- \( \text{hyperG0} \)
  prior mixing distribution.
- \( a \)
  shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is \( 0.0001 \).
- \( b \)
  scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is \( 0.0001 \). If \( 0 \), then the concentration is fixed set to \( a \).
- \( N \)
  number of MCMC iterations.
- \( \text{doPlot} \)
  logical flag indicating whether to plot MCMC iteration or not. Default to \( \text{TRUE} \).
- \( \text{nbclust_init} \)
  number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).
- \( \text{plotevery} \)
  an integer indicating the interval between plotted iterations when \( \text{doPlot} \) is \( \text{TRUE} \).
- \( \text{diagVar} \)
  logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is \( \text{TRUE} \) (diagonal variance).
DPMGibbsN

use_variance_hyperprior

logical flag indicating whether a hyperprior is added for the variance parameter.
Default is TRUE which decrease the impact of the variance prior on the posterior.
FALSE is useful for using an informative prior.

verbose

logical flag indicating whether partition info is written in the console at each
MCMC iteration.

additional arguments to be passed to plot_DPM. Only used if doPlot is TRUE.

Value

a object of class DPMclust with the following attributes:

mcmc_partitions:

a list of length N. Each element mcmc_partitions[n] is a vector of length n
giving the partition of the n observations.

alpha:

a vector of length N. cost[j] is the cost associated to partition c[[j]].

listU_mu:

a list of length N containing the matrices of mean vectors for all the mixture
components at each MCMC iteration

listU_Sigma:

a list of length N containing the arrays of covariances matrices for all the mixture
components at each MCMC iteration

U_SS_list:

a list of length N containing the lists of sufficient statistics for all the mixture
components at each MCMC iteration

weights_list:

a list of length N containing the logposterior values at each MCMC iterations

logposterior_list:

a list of length N containing the logposterior values at each MCMC iterations

data:

the data matrix d x n with d dimensions in rows and n observations in columns.

nb_mcmcit:

the number of MCMC iterations

clust_distrib:

the parametric distribution of the mixture component - "gaussian"

hyperG0:

the prior on the cluster location

Author(s)

Boris Hejblum

Examples

rm(list=ls())
#Number of data
n <- 500
d <- 4
#n <- 2000
set.seed(1234)
#set.seed(123)
#set.seed(4321)

# Sample data
m <- matrix(nrow=d, ncol=4, c(-1, 1, 1.5, 2, -2, -1.5, -2))
p <- c(0.2, 0.1, 0.4, 0.3) # frequente des clusters

sdev <- array(dim=c(d,d,4))
sdev[, ,1] <- 0.3*diag(d)
sdev[, ,2] <- c(0.1, 0.3)*diag(d)
sdev[, ,3] <- matrix(nrow=d, ncol=d, 0.15)
diag(sdev[, ,3]) <- 0.3
sdev[, ,4] <- 0.3*diag(d)
c <- rep(0,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
z[,k] <- m[, c[k]] + sdev[, , c[k]]%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
#cat(k, "/", n, " observations simulated\n", sep="")
}

# Set parameters of G0
hyperG0 <- list()
hyperG0[['mu']] <- rep(0,d)
hyperG0[['kappa']] <- 0.001
hyperG0[['nu']] <- d+2
hyperG0[['lambda']] <- diag(d)/10

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

# Number of iterations
N <- 30

# do some plots
doPlot <- TRUE
nbclust_init <- 30

### Data
library(ggplot2)
p <- (ggplot(data.frame("X"=z[1,], "Y"=z[2,]), aes(x=X, y=Y))
   + geom_point()
   + ggtitle("Toy example Data"))

### alpha priors plots
prioralpha <- data.frame("alpha"=rgamma(n=5000, shape=a, scale=1/b),
   "distribution" =factor(rep("prior",5000),
   levels=c("prior", "posterior")))
p <- (ggplot(prioralpha, aes(x=alpha))
   + geom_histogram(aes(y=..density..),
   colour="black", fill="white", bins=30)
DPMGibbsN

```r
+ geom_density(alpha=.6, fill="red", color=NA)
+ ggtitle(paste("Prior distribution on alpha: Gamma("a, ", b ", ")\n", sep=""))
+ theme_bw()

p

if(interactive()){
# Gibbs sampler for Dirichlet Process Mixtures

MCMCsample <- DPMGibbsN(z, hyperG0, a, b, N=500, doPlot, nbclust_init, plotevery=100,
    gg.add=list(theme_bw(),
    guides(shape=guide_legend(override.aes = list(fill="grey45"))),
    diagVar=FALSE)

plot_ConvDPM(MCMCsample, from=2)

s <- summary(MCMCsample, burnin = 200, thin=2, posterior_approx=FALSE,
    lossFn = "MBinderN")

F <- FmeasureC(pred=s$point_estim$c_est, ref=c)

postalpha <- data.frame("alpha"=MCMCsample$alpha[50:500],
    "distribution" = factor(rep("posterior",500-49),
    levels=c("prior", "posterior")))
p <- (ggplot(postalpha, aes(x=alpha))
    + geom_histogram(aes(y=..density..), binwidth=.1,
        colour="black", fill="white")
    + geom_density(alpha=.2, fill="blue")
    + ggtitle("Posterior distribution of alpha\n")
    # Ignore NA values for mean
    # Overlay with transparent density plot
    + geom_vline(aes(xintercept=mean(alpha, na.rm=TRUE)),
        color="red", linetype="dashed", size=1)
)

p

p <- (ggplot(drop=FALSE, alpha=.6)
    + geom_density(aes(x=alpha, fill=distribution),
        color=NA, alpha=.6,
        data=prioralpha)
    + geom_density(aes(x=alpha, fill=distribution),
        color=NA, alpha=.6,
        data=postalpha)
    + ggtitle("Prior and posterior distributions of alpha\n")
    + scale_fill_discrete(drop=FALSE)
    + theme_bw()
    +xlim(0,10)
    +ylim(0, 1.3)
)

p
```
DPMGibbsN_parallel

Slice Sampling of the Dirichlet Process Mixture Model with a prior on alpha

Description

Slice Sampling of the Dirichlet Process Mixture Model with a prior on alpha

Usage

DPMGibbsN_parallel(
  Ncpus,
  type_connec,
  z,
  hyperG0,
  a = 1e-04,
  b = 1e-04,
  N,
  doPlot = TRUE,
  nbclust_init = 30,
  plotevery = N/10,
  ...)

# k-means comparison

plot(x=z[,1], y=z[,2], col=kmeans(t(z), centers=4)$cluster,
     xlab = "d = 1", ylab= "d = 2", main="k-means with K=4 clusters")

KM <- kmeans(t(z), centers=4)
dataKM <- data.frame("X"=z[,1], "Y"=z[,2],
                   "Cluster"=as.character(KM$cluster))
dataCenters <- data.frame("X"=KM$centers[,1],
                        "Y"=KM$centers[,2],
                        "Cluster"=rownames(KM$centers))

p <- (ggplot(dataKM)
     + geom_point(aes(x=X, y=Y, col=Cluster))
     + geom_point(aes(x=X, y=Y, fill=Cluster, order=Cluster),
                  data=dataCenters, shape=22, size=5)
     + scale_colour_discrete(name="Cluster")
     + ggtitle("K-means with K=4 clusters\n"))

p
diagVar = TRUE,
use_variance_hyperprior = TRUE,
verbose = TRUE,
monitorfile = "",
... )

Arguments

Ncpus  the number of processors available

type_connec  The type of connection between the processors. Supported cluster types are
"SOCK", "FORK", "MPI", and "NWS". See also makeCluster.

z  data matrix $d \times n$ with $d$ dimensions in rows and $n$ observations in columns.

hyperGo  prior mixing distribution.

a  shape hyperparameter of the Gamma prior on the concentration parameter of the
Dirichlet Process. Default is 0.0001.

b  scale hyperparameter of the Gamma prior on the concentration parameter of the
Dirichlet Process. Default is 0.0001. If 0, then the concentration is fixed set to
a.

N  number of MCMC iterations.

doPlot  logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.

nbclust_init  number of clusters at initialization. Default to 30 (or less if there are less than
30 observations).

plotevery  an integer indicating the interval between plotted iterations when doPlot is
TRUE.

diagVar  logical flag indicating whether the variance of each cluster is estimated as a
diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).

use_variance_hyperprior  logical flag indicating whether a hyperprior is added for the variance parameter.
Default is TRUE which decrease the impact of the variance prior on the posterior.
FALSE is useful for using an informative prior.

verbose  logical flag indicating whether partition info is written in the console at each
MCMC iteration.

monitorfile  a writable connections or a character string naming a file to write into, to monitor
the progress of the analysis. Default is "" which is no monitoring. See Details.

...  additional arguments to be passed to plot_DPM. Only used if doPlot is TRUE.

Value

a object of class DPMclust with the following attributes:

mcmc_partitions:

a list of length $N$. Each element mcmc_partitions[n] is a vector of length $n$
giving the partition of the $n$ observations.

alpha:  a vector of length $N$. cost[j] is the cost associated to partition c[[j]]
listU_mu: a list of length \( N \) containing the matrices of mean vectors for all the mixture components at each MCMC iteration

listU_Sigma: a list of length \( N \) containing the arrays of covariances matrices for all the mixture components at each MCMC iteration

U_SS_list: a list of length \( N \) containing the lists of sufficient statistics for all the mixture components at each MCMC iteration

weights_list: a list of length \( N \) containing the logposterior values at each MCMC iterations

logposterior_list: a list of length \( N \) containing the logposterior values at each MCMC iterations

data: the data matrix \( d \times n \) with \( d \) dimensions in rows and \( n \) observations in columns

nb_mcmcита: the number of MCMC iterations

clust_distrib: the parametric distribution of the mixture component - "gaussian"

hyperG0: the prior on the cluster location

Author(s)
Boris Hejblum

See Also
DPMGibbsN

Examples

```r
# Scaling up: ----
rm(list=ls())
#Number of data
n <- 2000
set.seed(1234)

# Sample data
d <- 3
nclust <- 5
m <- matrix(nrow=d, ncol=nclust, runif(d*nclust)*8)
# p: cluster probabilities
p <- runif(nclust)
p <- p/sum(p)

# Covariance matrix of the clusters
sdev <- array(dim=c(d, d, nclust))
for (j in 1:nclust){
  sdev[, ,j] <- matrix(NA, nrow=d, ncol=d)
  diag(sdev[, ,j]) <- abs(rnorm(n=d, mean=0.3, sd=0.1))
  sdev[, ,j][lower.tri(sdev[, ,j], diag = FALSE)] <- rnorm(n=d*(d-1)/2, mean=0, sd=0.05)
  sdev[, ,j][upper.tri(sdev[, ,j], diag = FALSE)] <- (sdev[, ,j][lower.tri(sdev[, ,j], diag = FALSE)]
}
```
DPM Gibbs N_SeqPrior

Slice Sampling of Dirichlet Process Mixture of Gaussian distributions

Description

Slice Sampling of Dirichlet Process Mixture of Gaussian distributions

Usage

DPMGibbsN_SeqPrior(
DPMGibbsN_SeqPrior

z,
prior_inform,
hyperG0,
N,
nbclust_init,
add.vagueprior = TRUE,
weightnoninfo = NULL,
doPlot = TRUE,
plotevery = N/10,
diagVar = TRUE,
verbose = TRUE,
...
)

Arguments

z data matrix d x n with d dimensions in rows and n observations in columns.
prior_inform an informative prior such as the approximation computed by summary.DPMMclust.
hyperG0 a non informative prior component for the mixing distribution. Only used if add.vagueprior is TRUE.
N number of MCMC iterations.
nbclust_init number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).
add.vagueprior logical flag indicating whether a non informative component should be added to the informative prior. Default is TRUE.
weightnoninfo a real between 0 and 1 giving the weights of the non informative component in the prior.
doPlot logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.
plotevery an integer indicating the interval between plotted iterations when doPlot is TRUE.
diagVar logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).
verbose logical flag indicating whether partition info is written in the console at each MCMC iteration.
... additional arguments to be passed to plot_DPM. Only used if doPlot is TRUE.

Value

a object of class DPMMclust with the following attributes:

mcmc_partitions: a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.
alpha: a vector of length N. cost[j] is the cost associated to partition c[[j]]
listU_mu: a list of length N containing the matrices of mean vectors for all the mixture components at each MCMC iteration
listU_Sigma: a list of length N containing the arrays of covariances matrices for all the mixture components at each MCMC iteration
U_SS_list: a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration
weights_list: logposterior_list: a list of length N containing the logposterior values at each MCMC iterations
data: the data matrix d x n with d dimensions in rows and n observations in columns.
nb_mcmcit: the number of MCMC iterations
clust_distrib: the parametric distribution of the mixture component - "gaussian"
hyperG0: the prior on the cluster location

Author(s)
Boris Hejblum, Chariff Alkhassim

References
doi:10.1214/18AOAS1209

See Also
postProcess.DPMMclust DPMGibbsN

Examples

```r
rm(list=ls())
library(NPflow)
#Number of data
n <- 1500
# Sample data
#m <- matrix(nrow=2, ncol=4, c(-1, 1, 1.5, 2, 2, -2, 0.5, -2))
m <- matrix(nrow=2, ncol=4, c(-.8, .7, .5, .7, .5, -.7, -.5, -.7))
p <- c(0.2, 0.1, 0.4, 0.3) # frequence des clusters
sdev <- array(dim=c(2,2,4))
sdev[, ,1] <- matrix(nrow=2, ncol=2, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=2, ncol=2, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=2, ncol=2, c(0.3, 0.15, 0.15, 0.3))
sdev[, ,4] <- .3*diag(2)
c <- rep(0,n)
z <- matrix(0, nrow=2, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!)=0)
z[,k] <- m[, c[k]] + sdev[, , c[k]]%*%matrix(rnorm(2, mean = 0, sd = 1), nrow=2, ncol=1)
}
```
# DPMGibbsN_SeqPrior

#cat(k, "/", n, " observations simulated\n", sep="")
}

d<-2
# Set parameters of G0
hyperG0 <- list()
hyperG0["mu"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["nu"] <- d+2
hyperG0["lambda"] <- diag(d)/10

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

# Number of iterations
N <- 30

# do some plots
doPlot <- TRUE
nbclust_init <- 20

### Data
library(ggplot2)
p <- (ggplot(data.frame("X"=z[1,], "Y"=z[2,]), aes(x=X, y=Y))
  + geom_point()
  + ggtitle("Toy example Data"))
p

if(interactive()){
  # Gibbs sampler for Dirichlet Process Mixtures

  MCMCSample <- DPMGibbsN(z, hyperG0, a, b, N=1500, doPlot, nbclust_init, plotevery=200,
    gg.add=list(theme_bw(),
      guides(shape=guide_legend(override.aes = list(fill="grey45"))),
      diagVar=FALSE)

  s <- summary(MCMCSample, posterior_approx=TRUE, burnin = 1000, thin=5)
  F1 <- FmeasureC(pred=s$point_estim$c_est, ref=c)
  F1

  MCMCSample2 <- DPMGibbsN_SeqPrior(z, prior_inform=s$param_posterior,
    hyperG0, N=1500,
    add.vagueprior = TRUE,
    doPlot=TRUE, plotevery=100,
    nbclust_init=nbclust_init,
    gg.add=list(theme_bw(),
      guides(shape=guide_legend(override.aes = list(fill="grey45")))
    )

Slice Sampling of Dirichlet Process Mixture of skew normal distributions

**Description**

Slice Sampling of Dirichlet Process Mixture of skew normal distributions

**Usage**

```r
DPMGibbsSkewN(
  z,
  hyperG0,
  a = 1e-04,
  b = 1e-04,
  N,
  doPlot = TRUE,
  nbclust_init = 30,
  plotevery = N/10,
  diagVar = TRUE,
  use_variance_hyperprior = TRUE,
  verbose = TRUE,
  ...
)
```

**Arguments**

- `z` data matrix $d \times n$ with $d$ dimensions in rows and $n$ observations in columns.
- `hyperG0` prior mixing distribution.
- `a` shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001.
- `b` scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001. If 0, then the concentration is fixed set to $a$.
- `N` number of MCMC iterations.
- `doPlot` logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.
DPMGibbsSkewN

nbclust_init number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).

plotevery an integer indicating the interval between plotted iterations when doPlot is TRUE.

diagVar logical flag indicating whether the variance of a cluster is a diagonal matrix. Default is FALSE (full matrix).

use_variance_hyperprior logical flag indicating whether a hyperprior is added for the variance parameter. Default is TRUE which decrease the impact of the variance prior on the posterior. FALSE is useful for using an informative prior.

verbose logical flag indicating whether partition info is written in the console at each MCMC iteration.

... additional arguments to be passed to plot_DPMsn. Only used if doPlot is TRUE.

Value

a object of class DPMclust with the following attributes:

mcmc_partitions: a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.

alpha: a vector of length N. cost[j] is the cost associated to partition c[[j]]

U_SS_list: a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration

weights_list:

logposterior_list: a list of length N containing the logposterior values at each MCMC iterations

data: the data matrix d x n with d dimensions in rows and n observations in columns

nb_mcmcit: the number of MCMC iterations

clust_distrib: the parametric distribution of the mixture component - "skewnorm"

hyperG0: the prior on the cluster location

Author(s)

Boris Hejblum

References

Examples

```r
rm(list=ls())

# Number of data
n <- 1000
set.seed(123)

d <- 2
ncl <- 4

# Sample data
sdev <- array(dim=c(d,d,ncl))

# xi <- matrix(nrow=d, ncol=ncl, c(-1.5, 1, 1.5, 1, 1.5, -2, -2, -2))
xi <- matrix(nrow=d, ncol=ncl, c(-0.5, 0, 0.5, 0, 0.5, -1, -1, 1))

# psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -1.2))
psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -1.2))

p <- c(0.2, 0.1, 0.4, 0.3) # frequency of clusters

sdev[, ,1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[, ,4] <- .3*diag(2)

c <- rep(0,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
  c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
  z[,k] <- xi[, c[k]] + psi[, c[k]]*abs(rnorm(1)) + sdev[, , c[k]]*matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
  #cat(k, "/", n, " observations simulated\n", sep="")
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rep(0,d)
hyperG0["b_psi"] <- rep(0,d)
hyperG0["kappa"] <- 0.0001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d + 1
hyperG0["lambda"] <- diag(d)

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

# do some plots
doPlot <- TRUE
nbclust_init <- 30
```
## Data

```r
library(ggplot2)
p <- ggplot(data.frame("X"=z[1,], "Y"=z[2,], aes(x=X, y=Y))
+ geom_point()
+ ggtitle("Simple example in 2d data")
+xlab("D1")
+ylab("D2")
+theme_bw())
p
c2plot <- factor(c)
levels(c2plot) <- c("3", "2", "4", "1")
pp <- ggplot(data.frame("X"=z[1,], "Y"=z[2,], "Cluster"=as.character(c2plot)))
+ geom_point(aes(x=X, y=Y, colour=Cluster, fill=Cluster))
+ ggtitle("Slightly overlapping skew-normal simulation\n")
+xlab("D1")
+ylab("D2")
+ theme_bw()
+ scale_colour_discrete(guide=guide_legend(override.aes = list(size = 6, shape=22))))
pp

## alpha priors plots

```r
prioralpha <- data.frame("alpha"=rgamma(n=5000, shape=a, scale=1/b),
  "distribution" =factor(rep("prior",5000),
  levels=c("prior", "posterior")))
p <- ggplot(prioralpha, aes(x=alpha))
+ geom_histogram(aes(y=..density..),
  colour="black", fill="white")
+ geom_density(alpha=.2, fill="red")
+ ggtitle(paste("Prior distribution on alpha: Gamma(", a,
  ",", b, ")\n", sep=""))
)
p
if(interactive()){  
  # Gibbs sampler for Dirichlet Process Mixtures
  MCMCsample_sn <- DPMGibbsSkewN(z, hyperG0, a, b, N=2500,
  doPlot, nbclust_init, plotevery=200,
  gg.add=list(theme_bw(),
guides(shape=guide_legend(override.aes = list(fill="grey45"))))),
  diagVar=FALSE)
s <- summary(MCMCsample_sn, burnin = 2000, thin=10)
  #cluster_est_binder(MCMCsample_sn$mcmc_partitions[1000:1500])
  print(s)
}
plot(s)
#plot_ConvDPM(MCMCsamp;le_sn, from=2)

# k-means

plot(x=z[,1], y=z[,2], col=kmeans(t(z), centers=4)$cluster,
     xlab = "d = 1", ylab = "d = 2", main="k-means with K=4 clusters")

KM <- kmeans(t(z), centers=4)
KMclust <- factor(KM$cluster)
levels(KMclust) <- c("2", "4", "1", "3")
dataKM <- data.frame("X"=z[,1], "Y"=z[,2],
                    "Cluster"=as.character(KMclust))
dataCenters <- data.frame("X"=KM$centers[,1],
                           "Y"=KM$centers[,2],
                           "Cluster"=c("2", "4", "1", "3"))

p <- (ggplot(dataKM)
   + geom_point(aes(x=X, y=Y, col=Cluster))
   + geom_point(aes(x=X, y=Y, fill=Cluster, order=Cluster),
                data=dataCenters, shape=22, size=5)
   + scale_colour_discrete(name="Cluster",
                           guide=guide_legend(override.aes=list(size=6, shape=22)))
   + ggtitle("K-means with K=4 clusters\n")
   + theme_bw())

postalpha <- data.frame("alpha"=MCMCsamp;le_sn$alpha[501:1000],
                       "distribution" = factor(rep("posterior",1000-500),
                       levels=c("prior", "posterior")))

postK <- data.frame("K"=sapply(lapply(postalpha$alpha, ","),
                        function(x){sum(x/(x+0:(1000-1))}))

p <- (ggplot(postalpha, aes(x=alpha))
     + geom_histogram(aes(y=..density..), binwidth=.1,
                      colour="black", fill="white")
     + geom_density(alpha=.2, fill="blue")
     + ggtitle("Posterior distribution of alpha\n")
     + theme_bw())

p
p <- (ggplot(postK, aes(x=K))
  + geom_histogram(aes(y=..density..),
    colour="black", fill="white")
  + geom_density(alpha=2, fill="blue")
  + ggtitle("Posterior distribution of predicted K\n")
  # Ignore NA values for mean
  # Overlay with transparent density plot
  + geom_vline(aes(xintercept=mean(K, na.rm=T)),
    color="red", linetype="dashed", size=1)
  #+ scale_x_continuous(breaks=c(0:6)*2, minor_breaks=c(0:6)*2+1)
  + scale_x_continuous(breaks=c(1:12))
)
p
p <- (ggplot(drop=FALSE, alpha=.6)
  + geom_density(aes(x=alpha, fill=distribution),
    color=NA, alpha=.6,
    data=postalpha)
  + geom_density(aes(x=alpha, fill=distribution),
    color=NA, alpha=.6,
    data=prioralpha)
  + ggtitle("Prior and posterior distributions of alpha\n")
  + scale_fill_discrete(drop=FALSE)
  + theme_bw()
  + xlim(0,100)
)
p
#Skew Normal
n=100000
xi <- 0
d <- 0.995
alpha <- d/sqrt(1-d^2)
z <- rtruncnorm(n,a=0, b=Inf)
e <- rnorm(n, mean = 0, sd = 1)
x <- d*z + sqrt(1-d^2)*e
o <- 1
y <- xi+o*x
nu=1.3
w <- rgamma(n, scale=nu/2, shape=nu/2)
yy <- xi+o*x/w
snd <- data.frame("Y"=y,"YY"=yy)
p <- (ggplot(snd)+geom_density(aes(x=Y), fill="blue", alpha=.2)
  + theme_bw()
  + ylab("Density")
  + ggtitle("Y~SN(0,1,10)\n")
  + xlim(-1,6)
  + ylim(0,0.8)
)
p
p <- (ggplot(snd)+geom_density(aes(x=YY), fill="blue", alpha=.2)
  + theme_bw()
p <- ggplot(snd)
+ geom_density(aes(x=X, fill="blue"), alpha=.2)
+ theme_bw()
+ theme(legend.text = element_text(size = 13), legend.position="bottom")
+ ylab("Density")
+ ggtitle("X~SN(0,1,10)\n")
+ xlim(-1.5,4)
+ ylim(0,1.6)
)
p

o <- 0.5
y <- xi+o*x
snd <- data.frame("Y"=y)
p <- (ggplot(snd)+geom_density(aes(x=Y), fill="blue", alpha=.2)
+ theme_bw()
+ ylab("Density")
+ ggtitle("Y~SN(-1,1,10)\n")
+ xlim(-1.5,4)
+ ylim(0,1.6)
)
# Simple toy example

```r
n <- 500
set.seed(12345)

d <- 2
nc1 <- 4

# Sample data

sdev <- array(dim=c(d,d,nc1))

xi <- matrix(nrow=d, ncol=nc1, c(-1.5, 1.5, 1, 1.5, 1, 1.5, -2, -2, -2))
psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -1.2))
p <- c(0.2, 0.1, 0.4, 0.3) # frequences des clusters

sdev[, , 1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, , 2] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, , 3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[, , 4] <- .3*diag(2)

# Set parameters of G0

hyperG0 <- list()

hyperG0["b_xi"] <- rep(0, d)
hyperG0["b_psi"] <- rep(0, d)
hyperG0["kappa"] <- 0.0001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d + 1
hyperG0["lambda"] <- diag(d)

c <- rep(0, n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
  c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
  z[,k] <- xi[, c[k]] + psi[, c[k]]*abs(rnorm(1)) + sdev[, , c[k]]%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
}
cat(k, "/", n, " observations simulated\n", sep="")

MCMCsample_sn_sep <- DPMGibbsSkewN(z, hyperG0, a, b, N=600,
doPlot, nbclust_init, plotevery=100,
gg.add=list(theme_bw(),
guides(shape=guide_legend(override.aes = list(fill="grey45")))),
diagVar=TRUE)

s <- summary(MCMCsample_sn, burnin = 400)

}```
Description

If the `monitorfile` argument is a character string naming a file to write into, in the case of a new file that does not exist yet, such a new file will be created. A line is written at each MCMC iteration.

Usage

```r
DPMGibbsSkewN_parallel(
  Ncpus,
  type_connec,
  z,
  hyperG0,
  a = 1e-04,
  b = 1e-04,
  N,
  doPlot = FALSE,
  nbclust_init = 30,
  plotevery = N/10,
  diagVar = TRUE,
  use_variance_hyperprior = TRUE,
  verbose = FALSE,
  monitorfile = "",
  ...
)
```

Arguments

- **Ncpus**: the number of processors available
- **type_connec**: The type of connection between the processors. Supported cluster types are "SOCK", "FORK", "MPI", and "NWS". See also `makeCluster`.
- **z**: data matrix \( d \times n \) with \( d \) dimensions in rows and \( n \) observations in columns.
- **hyperG0**: prior mixing distribution.
- **a**: shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001.
- **b**: scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001. If 0, then the concentration is fixed set to \( a \).
- **N**: number of MCMC iterations.
- **doPlot**: logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.
nbclust_init  number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).

plotevery  an integer indicating the interval between plotted iterations when doPlot is TRUE.

diagVar  logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).

use_variance_hyperprior  logical flag indicating whether a hyperprior is added for the variance parameter. Default is TRUE which decrease the impact of the variance prior on the posterior. FALSE is useful for using an informative prior.

verbose  logical flag indicating whether partition info is written in the console at each MCMC iteration.

monitorfile  a writable connections or a character string naming a file to write into, to monitor the progress of the analysis. Default is "" which is no monitoring. See Details.

...  additional arguments to be passed to plot_DPM. Only used if doPlot is TRUE.

Value

a object of class DPMclust with the following attributes:

mcmc_partitions:  a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.

alpha:  a vector of length N. cost[j] is the cost associated to partition c[[j]]

U_SS_list:  a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration

weights_list:  logposterior_list:  a list of length N containing the logposterior values at each MCMC iterations

data:  the data matrix d x n with d dimensions in rows and n observations in columns

nb_mcmcit:  the number of MCMC iterations

clust_distrib:  the parametric distribution of the mixture component - "skewnorm"

hyperG0:  the prior on the cluster location

Author(s)

Boris Hejblum

References

Examples

```r
rm(list=ls())
# Number of data
n <- 2000
set.seed(1234)

d <- 4
ncl <- 5

# Sample data
sdev <- array(dim=c(d,d,ncl))
xi <- matrix(nrow=d, ncol=ncl, c(runif(n=d*ncl, min=0, max=3)))
psi <- matrix(nrow=d, ncol=ncl, c(runif(n=d*ncl, min=-1, max=1)))
p <- runif(n=ncl)
p <- p/sum(p)
sdev0 <- diag(runif(n=d, min=0.05, max=0.6))
for (j in 1:ncl){
  sdev[, ,j] <- invwishrnd(n = d+2, lambda = sdev0)
}
c <- rep(0,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
z[,k] <- xi[, c[k]] + psi[, c[k]]*abs(rnorm(1)) + sdev[, , c[k]]%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
  #cat(k, "/", n, " observations simulated\n", sep="")
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rep(0,d)
hyperG0["b_psi"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d + 1
hyperG0["lambda"] <- diag(d)/10

  # hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

  # do some plots
doPlot <- TRUE
nbclust_init <- 30

z <- z*200
```
### Data

```r
library(ggplot2)
p <- (ggplot(data.frame("X"=z[,1], "Y"=z[,2]), aes(x=X, y=Y))
  + geom_point()
  + ggtitle("Simple example in 2d data")
  + xlab("D1")
  + ylab("D2")
  + theme_bw())
p
## alpha priors plots

```r
prioralpha <- data.frame("alpha"=rgamma(n=5000, shape=a, scale=1/b),
  "distribution" =factor(rep("prior",5000),
  levels=c("prior", "posterior")))
p <- (ggplot(prioralpha, aes(x=alpha))
  + geom_histogram(aes(y=..density..),
    colour="black", fill="white")
  + geom_density(alpha=.2, fill="red")
  + ggtitle(paste("Prior distribution on alpha: Gamma(", a,
    ",", b, ")\n", sep=""))
  )
p
# Gibbs sampler for Dirichlet Process Mixtures

```r
if(interactive()){
  MCMCsample_sn_par <- DPMGibbsSkewN_parallel(Ncpus=parallel::detectCores()-1,
    type_connec="SOCK", z, hyperG0,
    a, b, N=5000, doPlot, nbclust_init,
    plotevery=25, gg.add=list(theme_bw(),
    guides(shape=guide_legend(override.aes = list(fill="grey45"))))
  plot_ConvDPM(MCMCsample_sn_par, from=2)
}
```
Usage

DPMGibbsSkewT(
    z,  
    hyperG0,  
    a = 1e-04,  
    b = 1e-04,  
    N,  
    doPlot = TRUE,  
    nbclust_init = 30,  
    plotevery = N/10,  
    diagVar = TRUE,  
    use_variance_hyperprior = TRUE,  
    verbose = TRUE,  
    ...
)

Arguments

z  
data matrix d x n with d dimensions in rows and n observations in columns.

hyperG0  
parameters of the prior mixing distribution in a list with the following named components:

- "b_xi": a vector of length d with the mean location prior parameter. Can be set as the empirical mean of the data in an Empirical Bayes fashion.
- "b_psi": a vector of length d with the skewness location prior parameter. Can be set as 0 a priori.
- "kappa": a strictly positive number part of the inverse-Wishart component of the prior on the variance matrix. Can be set as very small (e.g. 0.001) a priori.
- "D_xi": hyperprior controlling the information in $\xi$ (the larger the less information is carried). 100 is a reasonable value, based on Fruhwirth-Schnatter et al., Biostatistics, 2010.
- "D_psi": hyperprior controlling the information in $\psi$ (the larger the less information is carried). 100 is a reasonable value, based on Fruhwirth-Schnatter et al., Biostatistics, 2010
- "nu": a prior number on the degrees of freedom of the t component that must be strictly greater than d. Can be set as d + 1 for instance.
- "lambda": a d x d symmetric definitive positive matrix part of the inverse-Wishart component of the prior on the variance matrix. Can be set as the diagonal of empirical variance of the data in an Empirical Bayes fashion divided by a factor 3 according to Fruhwirth-Schnatter et al., Biostatistics, 2010.

a  
shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001.

b  
scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001. If 0, then the concentration is fixed set to a.
number of MCMC iterations.

**doPlot** logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.

**nbclust_init** number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).

**plotevery** an integer indicating the interval between plotted iterations when doPlot is TRUE.

**diagVar** logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).

**use_variance_hyperprior** logical flag indicating whether a hyperprior is added for the variance parameter. Default is TRUE which decrease the impact of the variance prior on the posterior. FALSE is useful for using an informative prior.

**verbose** logical flag indicating whether partition info is written in the console at each MCMC iteration.

... additional arguments to be passed to `plot_DPMst`. Only used if doPlot is TRUE.

### Value

A object of class DPMclust with the following attributes:

- **mcmc_partitions**: a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.
- **alpha**: a vector of length N. cost[j] is the cost associated to partition c[[]][j]
- **U_SS_list**: a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration
- **weights_list**: a list of length N containing the weights of each mixture component for each MCMC iteration
- **logposterior_list**: a list of length N containing the logposterior values at each MCMC iterations
- **data**: the data matrix d x n with d dimensions in rows and n observations in columns
- **nb_mcmcit**: the number of MCMC iterations
- **clust_distrib**: the parametric distribution of the mixture component - "skewt"
- **hyperG0**: the prior on the cluster location

### Author(s)

Boris Hejblum

### References


Examples

```r
rm(list=ls())

# Number of data
n <- 2000
set.seed(4321)

d <- 2
ncl <- 4

# Sample data
library(truncnorm)
sdev <- array(dim=c(d, d, ncl))

# xi <- matrix(nrow=d, ncol=ncl, c(-1.5, 1.5, 1.5, 1.5, 2, -2.5, -2.5, -3))
# xi <- matrix(nrow=d, ncol=ncl, c(-0.5, 0, 0.5, 0, 0.5, -1, -1, 1))
xi <- matrix(nrow=d, ncol=ncl, c(-0.2, 0.5, 2.4, 0.4, 0.6, -1.3, -0.9, -2.7))
psi <- matrix(nrow=d, ncol=4, c(0.3, -0.7, -0.8, 0, 0.3, -0.7, 0.2, 0.9))
nu <- c(100, 25, 8, 5)
p <- c(0.15, 0.05, 0.5, 0.3)  # frequency of clusters
sdev[, , 1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, , 2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[, , 3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[, , 4] <- 0.3*diag(2)
c <- rep(0, n)
w <- rep(1, n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
z[,k] <- xi[, c[k]] + psi[, c[k]]*rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
          (sdev[, , c[k]]/sqrt(w[k]))%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
#cat(k, ", ", n, " observations simulated\n", sep="")
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rowMeans(z)
hyperG0["b_psi"] <- rep(0, d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d+1
hyperG0["lambda"] <- diag(apply(z, MARGIN=1, FUN=var))/3

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001
```
## Data
###

```r
library(ggplot2)
p <- (ggplot(data.frame("X"=z[1,,], "Y"=z[2,,], aes(x=X, y=Y))
  + geom_point()
  + ggtitle("Simple example in 2d data")
  + xlab("D1")
  + ylab("D2")
  + theme_bw())
p #pdf(height=8.5, width=8.5)

c2plot <- factor(c)
levels(c2plot) <- c("4", "1", "3", "2")
pp <- (ggplot(data.frame("X"=z[1,,], "Y"=z[2,,], "Cluster"=as.character(c2plot)))
  + geom_point(aes(x=X, y=Y, colour=Cluster, fill=Cluster))
  + ggtitle("Slightly overlapping skew-normal simulation")
  + xlab("D1")
  + ylab("D2")
  + theme_bw()
  + scale_colour_discrete(guide=guide_legend(override.aes = list(size = 6, shape=22))))
pp #pdf(height=7, width=7.5)

## alpha priors plots
###

```r
prioralpha <- data.frame("alpha"=rgamma(n=5000, shape=a, scale=1/b),
  "distribution" =factor(rep("prior",5000),
  levels=c("prior", "posterior")))
p <- (ggplot(prioralpha, aes(x=alpha))
  + geom_histogram(aes(y=..density..),
    colour="black", fill="white")
  + geom_density(alpha=.2, fill="red")
  + ggtitle(paste("Prior distribution on alpha: Gamma(", a, ",", b, ")\n", sep="")))
p
```

if(interactive()){
  # Gibbs sampler for Dirichlet Process Mixtures
  MCMCsample_st <- DPMGibbsSkewT(z, hyperG0, a, b, N=1500,
    doPlot=TRUE, nbclust_init=30, plotevery=100,
    diagVar=FALSE)
  s <- summary(MCMCsample_st, burnin = 1000, thin=10, lossFn = "Binder")
  print(s)
  plot(s, hm=TRUE) #pdf(height=8.5, width=10.5) #png(height=700, width=720)
  plot_ConvDPM(MCMCsample_st, from=2)
  #cluster_est_binder(MCMCsample_st$mcmc_partitions[900:1000])

```
Slice Sampling of Dirichlet Process Mixture of skew Student’s t-distributions

Description

Slice Sampling of Dirichlet Process Mixture of skew Student’s t-distributions

Usage

DPMGibbsSkewT_parallel(
  Ncpus, type_connec, z, hyperG0, a = 1e-04, b = 1e-04, N,
  doPlot = FALSE, nbclust_init = 30, plotevery = N/10, diagVar = TRUE,
  use_variance_hyperprior = TRUE, verbose = FALSE,
  monitorfile = "", ...
)

Arguments

Ncpus the number of processors available

type_connec The type of connection between the processors. Supported cluster types are "PSOCK", "FORK", "SOCK", "MPI", and "NWS". See also makeCluster.

z data matrix d x n with d dimensions in rows and n observations in columns.

hyperG0 prior mixing distribution.

a shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001.

b scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001. If 0, then the concentration is fixed set to a.

N number of MCMC iterations.
doPlot logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.

nbclust_init number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).

plot.every an integer indicating the interval between plotted iterations when doPlot is TRUE.

diagVar logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).

use_variance_hyperprior logical flag indicating whether a hyperprior is added for the variance parameter. Default is TRUE which decrease the impact of the variance prior on the posterior. FALSE is useful for using an informative prior.

verbose logical flag indicating whether partition info is written in the console at each MCMC iteration.

monitorfile a writable connections or a character string naming a file to write into, to monitor the progress of the analysis. Default is "" which is no monitoring. See Details.

... additional arguments to be passed to plot_DPMst. Only used if doPlot is TRUE.

Value

a object of class DPMclust with the following attributes:

mcmc_partitions:
  a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.

alpha:
  a vector of length N. cost[j] is the cost associated to partition c[[j]]

U_SS_list:
  a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration

weights_list:
  a list of length N containing the weights of each mixture component for each MCMC iterations

logposterior_list:
  a list of length N containing the logposterior values at each MCMC iterations

data:
  the data matrix d x n with d dimensions in rows and n observations in columns

nb_mcmcit:
  the number of MCMC iterations

clust_distrib:
  the parametric distribution of the mixture component - "skewt"

hyperG0:
  the prior on the cluster location

Author(s)

Boris Hejblum

References

Examples

```
rm(list=ls())

# Number of data
n <- 2000
set.seed(123)
# set.seed(4321)

d <- 2
ncl <- 4

# Sample data
sdev <- array(dim=c(d,d,ncl))

xi <- matrix(nrow=d, ncol=ncl, c(-1.5, 1, 1.5, 1, 1.5, -2, -2, -2))
psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -0.8))
p <- c(0.2, 0.1, 0.4, 0.3)  # frequency of clusters
sdev[,1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[,2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[,3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[,4] <- .3*diag(2)

C <- rep(0,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
  C[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
  z[,k] <- (xi[, C[k]]
     + psi[, C[k]]*abs(rnorm(1))
     + sdev[, C[k]]*matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1))
  #cat(k, "/", n, " observations simulated\n", sep="")
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rep(0,d)
hyperG0["b_psi"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d + 1
hyperG0["lambda"] <- diag(d)

# hyperprior on the scale parameter of DPM
a <- 0.0001
b <- 0.0001

doPlot <- TRUE
nbclust_init <- 30
```
### Data

```r
library(ggplot2)
p <- ggplot(data.frame("X"=z[1,], "Y"=z[2,]), aes(x=X, y=Y))
  + geom_point()
  + ggtitle("Simple example in 2d data")
  + xlab("D1")
  + ylab("D2")
  + theme_bw()

p
c2plot <- factor(c)
levels(c2plot) <- c("3", "2", "4", "1")
pp <- ggplot(data.frame("X"=z[1,], "Y"=z[2,], "Cluster"=as.character(c2plot)))
  + geom_point(aes(x=X, y=Y, colour=Cluster, fill=Cluster))
  + ggtitle("Slightly overlapping skew-normal simulation")
  + xlab("D1")
  + ylab("D2")
  + theme_bw()
  + scale_colour_discrete(guide=guide_legend(override.aes = list(size = 6, shape=22))))

pp
```

### alpha priors plots

```r
prioralpha <- data.frame("alpha"=rgamma(n=5000, shape=a, scale=1/b),
  "distribution" =factor(rep("prior",5000),
  levels=c("prior", "posterior"))
p <- ggplot(prioralpha, aes(x=alpha))
  + geom_histogram(aes(y=..density..),
    colour="black", fill="white")
  + geom_density(alpha=.2, fill="red")
  + ggtitle(paste("Prior distribution on alpha: Gamma(", a,
    ",", b, ")\n", sep=""))

p
```

```r
if(interactive()){
    # Gibbs sampler for Dirichlet Process Mixtures
    MCMCsample_st <- DPMGibbsSkewT(z, hyperG0, a, b, N=2000,
      doPlot, nbclust_init, plotevery=100, gg.add=list(theme_bw(),
      guides(shape=guide_legend(override.aes = list(fill="grey45"))),
      diagVar=FALSE)
    s <- summary(MCMCsample_st, burnin = 350)
    print(s)
    plot(s)
    plot_ConvDPM(MCMCsample_st, from=2)
    cluster_est_binder(MCMCsample_st$mcmc_partitions[1500:2000])
}
### Description

Slice Sampling of Dirichlet Process Mixture of skew Student’s $t$-distributions

### Usage

```r
DPMGibbsSkewT_SeqPrior(
  z,
  prior_inform,
  hyperG0,
  N,
  nbclust_init,
  add.vagueprior = TRUE,
  weightnoninfo = NULL,
  doPlot = TRUE,
  plotevery = N/10,
  diagVar = TRUE,
  verbose = TRUE,
  ...
)
```

### Arguments

- **z**: data matrix $d \times n$ with $d$ dimensions in rows and $n$ observations in columns.
- **prior_inform**: an informative prior such as the approximation computed by `summary.DPMMclust`.
- **hyperG0**: prior mixing distribution.
- **N**: number of MCMC iterations.
- **nbclust_init**: number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).
- **add.vagueprior**: logical flag indicating whether a non-informative component should be added to the informative prior. Default is TRUE.
- **weightnoninfo**: a real between 0 and 1 giving the weights of the non-informative component in the prior.
- **doPlot**: logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.
plotevery an integer indicating the interval between plotted iterations when doPlot is TRUE.
diagVar logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).
verbose logical flag indicating whether partition info is written in the console at each MCMC iteration.
... additional arguments to be passed to plot_DPM. Only used if doPlot is TRUE.

Value

a object of class DPMclust with the following attributes:
mcmc_partitions: a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.
alpha: a vector of length N. cost[j] is the cost associated to partition c[[j]]
U_SS_list: a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration
weights_list: a list of length N containing the weights of each mixture component for each MCMC iteration
logposterior_list: a list of length N containing the logposterior values at each MCMC iterations
data: the data matrix d x n with d dimensions in rows and n observations in columns
nb_mcmcit: the number of MCMC iterations
clust_distrib: the parametric distribution of the mixture component - "skewt"
hyperG0: the prior on the cluster location

Author(s)
Boris Hejblum

References

Examples
rm(list=ls())

#Number of data
n <- 2000
set.seed(123)
d <- 2
ncl <- 4

# Sample data
sdev <- array(dim=c(d,d,ncl))

xi <- matrix(nrow=d, ncol=ncl, c(-1.5, 1.5, 1.5, 1.5, -2, -2, -2))
psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -0.8))
nu <- c(100,15,8,5)
p <- c(0.15, 0.05, 0.5, 0.3) # frequence des clusters
sdev[, ,1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[, ,4] <- .3*diag(2)

c <- rep(0,n)
w <- rep(1,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
z[,k] <- xi[, c[k]] + psi[, c[k]]*rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
(sdev[, , c[k]]/sqrt(w[k]))%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rowMeans(z)
hyperG0["b_psi"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d+1
hyperG0["lambda"] <- diag(apply(z,MARGIN=1, FUN=var))/3

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

# do some plots
nbclust_init <- 30

### Plot Data
library(ggplot2)
q <- (ggplot(data.frame("X"=z[1,], "Y"=z[2,]), aes(x=X, y=Y))
  + geom_point() 
  + ggtitle("Simple example in 2d data") 
  + xlab("D1") 
  + ylab("D2") 
  + theme_bw())
if(interactive()){
  MCMCsample_st <- DPMGibbsSkewT(z, hyperG0, a, b, N=2000,
                                 doPlot=TRUE, plotevery=250,
                                 nbclust_init, diagVar=FALSE,
                                 gg.add=list(theme_bw(),
                                 guides(shape=guide_legend(override.aes = list(fill="grey45")))))
  s <- summary(MCMCsample_st, burnin = 1500, thin=2, posterior_approx=TRUE)
  F <- FmeasureC(pred=s$point_estim$c_est, ref=c)
  for(k in 1:n){
    c[k] = which(rmultinom(n=1, size=1, prob=prob)!=0)
    w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
    z[,k] <- xi[, c[k]] + psi[, c[k]]*rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
             (sdev[, , c[k]]/sqrt(w[k]))*%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
    cat(k, "/", n, " observations simulated\n"
    }
    MCMCsample_st2 <- DPMGibbsSkewT_SeqPrior(z, prior=s$param_posterior,
                                              hyperG0, N=2000,
                                              doPlot=TRUE, plotevery=100,
                                              nbclust_init, diagVar=FALSE,
                                              gg.add=list(theme_bw(),
                                              guides(shape=guide_legend(override.aes = list(fill="grey45"))))
    s2 <- summary(MCMCsample_st2, burnin = 1500, thin=5)
    F2 <- FmeasureC(pred=s2$point_estim$c_est, ref=c)
  }
  MCMCsample_st2_par <- DPMGibbsSkewT_SeqPrior_parallel(Ncpus= 2, type_connec="SOCK",
                                                       z, prior_inform=s$param_posterior,
                                                       hyperG0, N=2000,
                                                       doPlot=TRUE, plotevery=50,
                                                       nbclust_init, diagVar=FALSE,
                                                       gg.add=list(theme_bw(),
                                                       guides(shape=guide_legend(override.aes = list(fill="grey45"))))
  }

# DPMGibbsSkewT_SeqPrior_parallel

Slice Sampling of Dirichlet Process Mixture of skew Student's t-distributions

Description

Slice Sampling of Dirichlet Process Mixture of skew Student's t-distributions

Usage

DPMGibbsSkewT_SeqPrior_parallel()}
Arguments

Ncpus

the number of processors available

type_connec

The type of connection between the processors. Supported cluster types are "SOCK", "FORK", "MPI", and "NWS". See also \texttt{makeCluster}.

z

data matrix \(d \times n\) with \(d\) dimensions in rows and \(n\) observations in columns.

prior_inform

an informative prior such as the approximation computed by \texttt{summary.DPMMclust}.

hyperG0

prior mixing distribution.

N

number of MCMC iterations.

nbclust_init

number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).

add.vagueprior

logical flag indicating whether a non informative component should be added to the informative prior. Default is \texttt{TRUE}.

weightnoninfo

a real between 0 and 1 giving the weights of the non informative component in the prior.

doPlot

logical flag indicating whether to plot MCMC iteration or not. Default to \texttt{TRUE}.

plotevery

an integer indicating the interval between plotted iterations when \texttt{doPlot} is \texttt{TRUE}.

diagVar

logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is \texttt{TRUE} (diagonal variance).

verbose

logical flag indicating whether partition info is written in the console at each MCMC iteration.

monitorfile

a writable \texttt{connections} or a character string naming a file to write into, to monitor the progress of the analysis. Default is \texttt{""} which is no monitoring. See Details.

... additional arguments to be passed to \texttt{plot_DPM}. Only used if \texttt{doPlot} is \texttt{TRUE}.
Value

A object of class DPMclust with the following attributes:

- `mcmc_partitions`: a list of length \( N \). Each element `mcmc_partitions[n]` is a vector of length \( n \) giving the partition of the \( n \) observations.
- `alpha`: a vector of length \( N \). `cost[j]` is the cost associated to partition \( c[[j]] \)
- `U_SS_list`: a list of length \( N \) containing the lists of sufficient statistics for all the mixture components at each MCMC iteration
- `weights_list`: a list of length \( N \) containing the logposterior values at each MCMC iteration
- `logposterior_list`: a list of length \( N \) containing the logposterior values at each MCMC iteration
- `data`: the data matrix \( d \times n \) with \( d \) dimensions in rows and \( n \) observations in columns
- `nb_mcmcit`: the number of MCMC iterations
- `clust_distrib`: the parametric distribution of the mixture component - "skewt"
- `hyperG0`: the prior on the cluster location

Author(s)

Boris Hejblum

References


Examples

```r
rm(list=ls())

# Number of data
n <- 2000
set.seed(123)

d <- 2
nc1 <- 4

# Sample data
sdev <- array(dim=c(d,d,nc1))

# xi <- matrix(nrow=d, ncol=nc1, c(-1.5, 1.5, 1.5, 1.5, 2, -2.5, -2.5, -3))
# psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -0.8))
xi <- matrix(nrow=d, ncol=nc1, c(-0.2, 0.5, 2.4, 0.4, 0.6, -1.3, -0.9, -2.7))
psi <- matrix(nrow=d, ncol=4, c(0.3, -0.7, -0.8, 0, 0.3, -0.7, 0.2, 0.9))
```
nu <- c(100, 15, 8, 5)
p <- c(0.15, 0.05, 0.5, 0.3)  # frequences des clusters
sdev[, , 1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, , 2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[, , 3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[, , 4] <- .3*diag(2)

c <- rep(0, n)
w <- rep(1, n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
z[k,] <- xi[, c[k]] + psi[, c[k]]*rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
(sdev[, , c[k]]/sqrt(w[k]))%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rowMeans(z)
hyperG0["b_psi"] <- rep(0, d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d+1
hyperG0["lambda"] <- diag(apply(z, MARGIN=1, FUN=var))/3

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

# do some plots
nbclust_init <- 30

## Plot Data
library(ggplot2)
q <- (ggplot(data.frame("X"=z[,1], "Y"=z[,2]), aes(x=X, y=Y))
+ geom_point()
+ ggtitle("Simple example in 2d data")
+xlab("D1")
+ylab("D2")
+theme_bw())
q

if(interactive()){
MCMCsample_st <- DPMGibbsSkewT(z, hyperG0, a, b, N=2000,
doPlot=TRUE, plotevery=250,
nbclust_init,
gg.add=list(theme_bw(),
guides(shape=guide_legend(override.aes = list(fill="grey45"))))))
}
DPMpost Posterior estimation for Dirichlet process mixture of multivariate (potentially skew) distributions models

Description

Partially collapse slice Gibbs sampling for Dirichlet process mixture of multivariate normal, skew normal or skew t distributions.

Usage

DPMpost(
  data,
  hyperG0,
  a = 1e-04,
  b = 1e-04,
  N,
  doPlot = TRUE,
  nbclust_init = 30,
  plotevery = floor(N/10),
  diagVar = TRUE,
  verbose = TRUE,
  distrib = c("gaussian", "skewnorm", "skewt"),
  ncores = 1,
  type_connec = "SOCK",
  informPrior = NULL,
)

s <- summary(MCMCsample_st, burnin = 1500, thin=5, posterior_approx=TRUE)
F <- FmeasureC(pred=s$point_estim$c_est, ref=c)

for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
z[,k] <- xi[, c[k]] + psi[, c[k]]*rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
    (sdev[, , c[k]]/sqrt(w[k]))%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
    #cat(k, "/", n, " observations simulated\n", sep="")
MCMCsample_st2 <- DPMGibbsSkewT_SeqPrior_parallel(Ncpus=2, type_connec="SOCK",
    z, prior_inform=s$param_posterior,
    hyperG0, N=3000,
    doPlot=TRUE, plotevery=100,
    nbclust_init, diagVar=FALSE, verbose=FALSE,
    gg.add=list(theme_bw(),
    guides(shape=guide_legend(override.aes = list(fill="grey45")))))
    s2 <- summary(MCMCsample_st2, burnin = 2000, thin=5)
F2 <- FmeasureC(pred=s2$point_estim$c_est, ref=c)
}
Arguments

data data matrix d x n with d dimensions in rows and n observations in columns.

hyperG0 prior mixing distribution.
a shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001.
b scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001. If 0, then the concentration is fixed set to a.
N number of MCMC iterations.
doPlot logical flag indicating whether to plot MCMC iteration or not. Default to TRUE.
nbclust_init number of clusters at initialization. Default to 30 (or less if there are less than 30 observations).
plotevery an integer indicating the interval between plotted iterations when doPlot is TRUE.
diagVar logical flag indicating whether the variance of each cluster is estimated as a diagonal matrix, or as a full matrix. Default is TRUE (diagonal variance).
verbose logical flag indicating whether partition info is written in the console at each MCMC iteration.
distrib the distribution used for the clustering. Current possibilities are "gaussian", "skewnorm" and "skewt".
ncores number of cores to use.
type_connec The type of connection between the processors. Supported cluster types are "SOCK", "FORK", "MPI", and "NWS". See also makeCluster.
informPrior an optional informative prior such as the approximation computed by summary.DPMMclust.
... additional arguments to be passed to plot_DPM. Only used if doPlot is TRUE.

Details

This function is a wrapper around the following functions: DPMGibbsN, DPMGibbsN_parallel, DPMGibbsN_SeqPrior, DPMGibbsSkewN, DPMGibbsSkewN_parallel, DPMGibbsSkewT, DPMGibbsSkewT_parallel, DPMGibbsSkewT_SeqPrior, DPMGibbsSkewT_SeqPrior_parallel.

Value

a object of class DPMclust with the following attributes:
mcmc_partitions a list of length N. Each element mcmc_partitions[n] is a vector of length n giving the partition of the n observations.
alpha: a vector of length N. cost[j] is the cost associated to partition c[[j]]
U_SS_list: a list of length N containing the lists of sufficient statistics for all the mixture components at each MCMC iteration
weights_list: a list of length N containing the weights of each mixture component for each MCMC iteration
logposterior_list: a list of length N containing the logposterior values at each MCMC iterations
data: the data matrix d x n with d dimensions in rows and n observations in columns
nb_mcmc: the number of MCMC iterations
clust_distrib: the parametric distribution of the mixture component
hyperG0: the prior on the cluster location

Author(s)
Boris Hejblum

References

See Also
summary.DPMMclust

Examples
#rm(list=ls())
set.seed(123)

# Exemple in 2 dimensions with skew-t distributions

# Generate data:
n <- 2000 # number of data points
d <- 2 # dimensions
ncl <- 4 # number of true clusters
sdev <- array(dim=c(d,d,ncl))
xi <- matrix(nrow=d, ncol=ncl, c(-1.5, 1.5, 1.5, 1.5, 2, -2.5, -2.5, -3))
psi <- matrix(nrow=d, ncol=4, c(0.3, -0.7, -0.8, 0, 0.3, -0.7, 0.2, 0.9))
nu <- c(100,25,8,5)
proba <- c(0.15, 0.05, 0.5, 0.3) # cluster frequencies
sdev[, ,1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.2))
sdev[, ,4] <- .3*diag(2)
c <- rep(0,n)
w <- rep(1,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
```r
c[k] = which(rmultinom(n=1, size=1, prob=proba)!=0)
w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
z[i,k] <- xi[, c[k]] + psi[, c[k]]*truncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
(sdev[, , c[k]]/sqrt(w[k]))%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
}

# Define hyperprior
hyperG0 <- list()
hyperG0["b_xi"] <- rowMeans(z)
hyperG0["b_psi"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d+1
hyperG0["lambda"] <- diag(apply(z,MARGIN=1, FUN=var))/3

if(interactive()){
  # Plot data
cytoScatter(z)

  # Estimate posterior
MCMCsample_st <- DPMpost(data=z, hyperG0=hyperG0, N=2000, distrib="skewt",
  distrib.add=list(ggplot2::theme_bw(),
    ggplot2::guides(shape=ggplot2::guide_legend(override.aes = list(fill="grey45"))))
  )
s <- summary(MCMCsample_st, burnin = 1600, thin=5, lossFn = "Binder")
s
plot(s)
#plot(s, hm=TRUE) # this can take a few sec...

  # more data plotting:
library(ggplot2)
p <- (ggplot(data.frame("X"=z[1,], "Y"=z[2,]), aes(x=X, y=Y))
  + geom_point()
  + ggtitle("Unsupervised data")
  + xlab("D1")
  + ylab("D2")
  + theme_bw()
  )
p
c2plot <- factor(c)
levels(c2plot) <- c("4", "1", "3", "2")
pp <- (ggplot(data.frame("X"=z[1,], "Y"=z[2,], "Cluster"=as.character(c2plot)))
  + geom_point(aes(x=X, y=Y, colour=Cluster, fill=Cluster))
  + ggtitle("True clusters")
  + xlab("D1")
  + ylab("D2")
  + theme_bw()
  + scale_colour_discrete(guide=guide_legend(override.aes = list(size = 6, shape=22)))
```

# Exemple in 2 dimensions with Gaussian distributions

```r
set.seed(1234)

# Generate data
n <- 2000  # number of data points
d <- 2  # # dimensions
ncl <- 4  # number of true clusters
m <- matrix(nrow=2, ncol=4, c(-1, 1, 1.5, 2, -2, -1.5, -2))  # cluster means
sdev <- array(dim=c(2, 2, 4))  # cluster standard-deviations
sdev[, ,1] <- matrix(nrow=2, ncol=2, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=2, ncol=2, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=2, ncol=2, c(0.3, 0.15, 0.15, 0.3))
sdev[, ,4] <- .3*diag(2)
proba <- c(0.15, 0.05, 0.5, 0.3)  # cluster frequencies
c <- rep(0,n)
z <- matrix(0, nrow=2, ncol=n)
for(k in 1:n){
  c[k] = which(rmultinom(n=1, size=1, prob=proba)!=0)
  z[,k] <- m[, c[k]] + sdev[, , c[k]]%*%matrix(rnorm(2, mean = 0, sd = 1), nrow=2, ncol=1)
}

# Define hyperprior
hyperG0 <- list()
hyperG0["mu"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["nu"] <- d+2
hyperG0["lambda"] <- diag(d)

if(interactive()){
  # Plot data
  cytoScatter(z)

  # Estimate posterior
  MCMCsample_n <- DPMpost(data=z, hyperG0=hyperG0, N=2000, distrib="gaussian", diagVar=FALSE, 
                          gg.add=list(ggplot2::theme_bw(), 
                                      ggplot2::guides(shape=ggplot2::guide_legend(override.aes = list(fill="grey45"))))
  )
  s <- summary(MCMCsample_n, burnin = 1500, thin=5, lossFn = "Binder")
  plot(s)
  #plot(s, hm=TRUE) # this can take a few sec...
```
evalClustLoss

E Loss of a partition point estimate compared to a gold standard

Description
Evaluate the loss of a point estimate of the partition compared to a gold standard according to a given loss function

Usage
evalClustLoss(c, gs, lossFn = "F-measure", a = 1, b = 1)

Arguments

- `c`: vector of length n containing the estimated partition of the n observations.
- `gs`: vector of length n containing the gold standard partition of the n observations.
- `lossFn`: character string specifying the loss function to be used. Either "F-measure" or "Binder" (see Details). Default is "F-measure".
- `a`: only relevant if `lossFn` is "Binder". Penalty for wrong co-clustering in `c` compared to `gs`. Defaults is 1.
- `b`: only relevant if `lossFn` is "Binder". Penalty for missed co-clustering in `c` compared to `gs`. Defaults is 1.
Details

The cost of a point estimate partition is calculated using either pairwise coincidence loss function (Binder), or 1-Fmeasure (F-measure).

Value

the cost of the point estimate \( c \) in regard of the gold standard \( g_s \) for a given loss function.

Author(s)

Boris Hejblum

References


See Also

similarityMat, cluster_est_binder

Flimited

Compute a limited F-measure

Description

A limited version of F-measure that only takes into account small clusters

Usage

Flimited(n_small_clst, pred, ref)

Arguments

n_small_clst an integer for limit size of the small cluster
pred vector of a predicted partition
ref vector of a reference partition

References

FmeasureC

Examples

```r
pred <- c(rep(1, 5), rep(2, 8), rep(3, 10))
ref <- c(rep(1, 5), rep(c(2, 3), 4), rep(c(3, 2), 5))
FmeasureC(pred, ref)
Flimited(6, pred, ref)
```

FmeasureC

C++ implementation of the F-measure computation

Description

C++ implementation of the F-measure computation

Usage

```r
FmeasureC(pred, ref)
```

Arguments

- `pred` vector of a predicted partition
- `ref` vector of a reference partition

Examples

```r
pred <- c(1, 1, 2, 3, 2)
ref <- c(2, 2, 1, 1, 3)
FmeasureC(pred, ref)
```

FmeasureC_no0

C++ implementation of the F-measure computation without the reference class 0

Description

Aghaeepour in FlowCAP 1 ignore the reference class labeled "0"

Usage

```r
FmeasureC_no0(pred, ref)
```

Arguments

- `pred` vector of a predicted partition
- `ref` vector of a reference partition
Multiple cost computations with the F-measure as the loss function

Description

C++ implementation of multiple cost computations with the F-measure as the loss function using the Armadillo library

Usage

Fmeasure_costC(c)

Arguments

c a matrix where each column is one MCMC partition

Value

a list with the following elements:

Fmeas: TODO

cost: TODO

Examples

library(NPflow)
c <- list(c(1,1,2,3,2,3), c(1,1,1,2,3,3), c(2,2,1,1,1,1))
Fmeasure_costC(sapply(c, "+")

if(interactive()){c2 <- list()for(i in 1:100){
c2 <- c(c2, list(rmultinom(n=1, size=2000, prob=rexp(n=2000))))
}Fmeasure_costC(sapply(c2, "+")

References


Examples

library(NPflow)
pred <- c(1,1,2,3,2,3)
ref <- c(2,2,0,0,0,3)
FmeasureC(pred, ref)
FmeasureC_no0(pred, ref)
Multivariate log gamma function

Description
Multivariate log gamma function

Usage
lgamma_mv(x, p)

Arguments
x
strictly positive real number
p
integer

EM MAP for mixture of sNiW

Description
Maximum A Posteriori (MAP) estimation of mixture of Normal inverse Wishart distributed observations with an EM algorithm

Usage
MAP_sNiW_mmEM(     xi_list,     psi_list,     S_list,     hyperG0,     init = NULL,     K,     maxit = 100,     tol = 0.1,     doPlot = TRUE,     verbose = TRUE     )

MAP_sNiW_mmEM_weighted(     xi_list,     psi_list,     ...
S_list,
obsweight_list,
hyperG0,
K,
maxit = 100,
tol = 0.1,
doPlot = TRUE,
verbose = TRUE
)

MAP_sNiW_mmEM_vague(
  xi_list,
  psi_list,
  S_list,
  hyperG0,
  K = 10,
  maxit = 100,
  tol = 0.1,
  doPlot = TRUE,
  verbose = TRUE
)

Arguments

xi_list a list of length n, each element is a vector of size d containing the argument \( x_i \) of the corresponding allocated cluster.

psi_list a list of length n, each element is a vector of size d containing the argument \( \psi_i \) of the corresponding allocated cluster.

S_list a list of length n, each element is a matrix of size \( d \times d \) containing the argument \( S \) of the corresponding allocated cluster.

hyperG0 prior mixing distribution used if \( \text{init} \) is \( \text{NULL} \).

init a list for initializing the algorithm with the following elements: \( \text{b_xi}, \text{b_psi}, \lambda, \text{B}, \nu \). Default is \( \text{NULL} \) in which case the initialization of the algorithm is random.

K integer giving the number of mixture components.

maxit integer giving the maximum number of iteration for the EM algorithm. Default is 100.

tol real number giving the tolerance for the stopping of the EM algorithm. Default is 0.1.

doPlot a logical flag indicating whether the algorithm progression should be plotted. Default is \( \text{TRUE} \).

verbose logical flag indicating whether plot should be drawn. Default is \( \text{TRUE} \).

obsweight_list a list of length n where each element is a vector of weights for each sampled cluster at each MCMC iterations.
Details

MAP_sNiW_mmEM provides an estimation for the MAP of mixtures of Normal inverse Wishart distributed observations. MAP_sNiW_mmEM_vague provides an estimates incorporating a vague component in the mixture. MAP_sNiW_mmEM_weighted provides a weighted version of the algorithm.

Author(s)

Boris Hejblum, Chariff Alkhassim

Examples

```r
set.seed(1234)
hyperG0 <- list()
hyperG0$b_xi <- c(0.3, -1.5)
hyperG0$b_psi <- c(0, 0)
hyperG0$kappa <- 0.001
hyperG0$D_xi <- 100
hyperG0$D_psi <- 100
hyperG0$nu <- 20
hyperG0$lambda <- diag(c(0.25, 0.35))

hyperG0 <- list()
hyperG0$b_xi <- c(1, -1.5)
hyperG0$b_psi <- c(0, 0)
hyperG0$kappa <- 0.1
hyperG0$D_xi <- 1
hyperG0$D_psi <- 1
hyperG0$nu <- 2
hyperG0$lambda <- diag(c(0.25, 0.35))

xi_list <- list()
psi_list <- list()
S_list <- list()
w_list <- list()

for(k in 1:200){
  NNiW <- rNNiW(hyperG0, diagVar=FALSE)
  xi_list[[k]] <- NNiW["xi"]
  psi_list[[k]] <- NNiW["psi"]
  S_list[[k]] <- NNiW["S"]
  w_list[[k]] <- 0.75
}

hyperG02 <- list()
hyperG02$b_xi <- c(-1, 2)
hyperG02$b_psi <- c(-0.1, 0.5)
hyperG02$kappa <- 0.1
hyperG02$D_xi <- 1
hyperG02$D_psi <- 1
hyperG02$nu <- 4
hyperG02$lambda <- 0.5*diag(2)
```
for(k in 201:400){
  NNiW <- rNNiW(hyperG02, diagVar=FALSE)
  xi_list[[k]] <- NNiW["xi"]
  psi_list[[k]] <- NNiW["psi"]
  S_list[[k]] <- NNiW["S"]
  w_list [[k]] <- 0.25
}
map <- MAP_sNiW_mmEM(xi_list, psi_list, S_list, hyperG0, K=2, tol=0.1)

---

**MLE\_gamma**  
**MLE for Gamma distribution**

**Description**

Maximum likelihood estimation of Gamma distributed observations distribution parameters

**Usage**

```r
MLE\_gamma(g)
```

**Arguments**

- **g**
  
a list of Gamma distributed observation.

**Examples**

```r
g_list <- list()
for(i in 1:1000){
  g_list <- c(g_list, rgamma(1, shape=100, rate=5))
}
mle <- MLE\_gamma(g_list)
mle
```
MLE_NiW_mmEM

Description

Maximum likelihood estimation of mixture of Normal inverse Wishart distributed observations with an EM algorithm

Usage

MLE_NiW_mmEM(
  mu_list,
  S_list,
  hyperG0,
  K,
  maxit = 100,
  tol = 0.1,
  doPlot = TRUE
)

Arguments

mu_list a list of length N whose elements are observed vectors of length d of the mean parameters.
S_list a list of length N whose elements are observed variance-covariance matrices of dimension d x d.
hyperG0 prior mixing distribution used for randomly initializing the algorithm.
K integer giving the number of mixture components.
maxit integer giving the maximum number of iteration for the EM algorithm. Default is 100.
tol real number giving the tolerance for the stopping of the EM algorithm. Default is 0.1.
doPlot a logical flag indicating whether the algorithm progression should be plotted. Default is TRUE.

Examples

set.seed(123)
U_mu <- list()
U_Sigma <- list()
U_nu <- list()
U_kappa <- list()
d <- 2
hyperG0 <- list()
hyperG0["mu"] <- rep(1,d)
MLE_sNiW

MLE for sNiW distributed observations

Description

Maximum likelihood estimation of Normal inverse Wishart distributed observations
Usage

MLE_sNiW(xi_list, psi_list, S_list, doPlot = TRUE)

Arguments

- **xi_list**: a list of length \(N\) whose elements are observed vectors of length \(d\) of the mean parameters \(xi\).
- **psi_list**: a list of length \(N\) whose elements are observed vectors of length \(d\) of the skew parameters \(psi\).
- **S_list**: a list of length \(N\) whose elements are observed variance-covariance matrices of dimension \(d \times d\).
- **doPlot**: a logical flag indicating whether the algorithm progression should be plotted. Default is TRUE.

Author(s)

Boris Hejblum, Chariff Alkhassim

Examples

```r
hyperG0 <- list()
hyperG0$b_xi <- c(0.3, -1.5)
hyperG0$b_psi <- c(0, 0)
hyperG0$kappa <- 0.001
hyperG0$D_xi <- 100
hyperG0$D_psi <- 100
hyperG0$nu <- 35
hyperG0$lambda <- diag(c(0.25, 0.35))

xi_list <- list()
psi_list <- list()
S_list <- list()
for(k in 1:1000){
    NNiW <- rNNiW(hyperG0, diagVar=FALSE)
    xi_list[[k]] <- NNiW["xi"]
    psi_list[[k]] <- NNiW["psi"]
    S_list[[k]] <- NNiW["S"]
}

mle <- MLE_sNiW(xi_list, psi_list, S_list)
print(mle)
```

Description

Maximum likelihood estimation of mixture of Normal inverse Wishart distributed observations with an EM algorithm.
Usage

MLE_sNiW_mmEM(
  xi_list,
  psi_list,
  S_list,
  hyperG0,
  K,
  init = NULL,
  maxit = 100,
  tol = 0.1,
  doPlot = TRUE,
  verbose = TRUE
)

Arguments

xi_list  a list of length \( N \) whose elements are observed vectors of length \( d \) of the mean parameters \( \xi \).

psi_list  a list of length \( N \) whose elements are observed vectors of length \( d \) of the skew parameters \( \psi \).

S_list  a list of length \( N \) whose elements are observed variance-covariance matrices of dimension \( d \times d \).

hyperG0  prior mixing distribution used if \( \text{init} \) is NULL.

K  integer giving the number of mixture components.

init  a list for initializing the algorithm with the following elements: \( b_{\xi}, b_{\psi}, \lambda, B, \nu \). Default is NULL in which case the initialization of the algorithm is random.

maxit  integer giving the maximum number of iteration for the EM algorithm. Default is 100.

tol  real number giving the tolerance for the stopping of the EM algorithm. Default is 0.1.

doPlot  a logical flag indicating whether the algorithm progression should be plotted. Default is TRUE.

verbose  logical flag indicating whether plot should be drawn. Default is TRUE.

Author(s)

Boris Hejblum, Chariff Alkhassim

Examples

set.seed(1234)
hyperG0 <- list()
hyperG0$b_xi <- c(0.3, -1.5)
hyperG0$b_psi <- c(0, 0)
hyperG0$kappa <- 0.001
hyperG0$D_xi <- 100
hyperG0$D_psi <- 100
hyperG0$nu <- 3
hyperG0$lambda <- diag(c(0.25, 0.35))

xi_list <- list()
psi_list <- list()
S_list <- list()
for(k in 1:200){
  NNiW <- rNNiW(hyperG0, diagVar=FALSE)
  xi_list[[k]] <- NNiW["xi"]
  psi_list[[k]] <- NNiW["psi"]
  S_list[[k]] <- NNiW["S"]
}

hyperG02 <- list()
hyperG02$b_xi <- c(-1, 2)
hyperG02$b_psi <- c(-0.1, 0.5)
hyperG02$kappa <- 0.001
hyperG02$D_xi <- 10
hyperG02$D_psi <- 10
hyperG02$nu <- 3
hyperG02$lambda <- 0.5*diag(2)
for(k in 201:400){
  NNiW <- rNNiW(hyperG02, diagVar=FALSE)
  xi_list[[k]] <- NNiW["xi"]
  psi_list[[k]] <- NNiW["psi"]
  S_list[[k]] <- NNiW["S"]
}

mle <- MLE_sNiW_mmEM(xi_list, psi_list, S_list, hyperG0, K=2)

mmNiWpdf

multivariate Normal inverse Wishart probability density function for multiple inputs

Description

multivariate Normal inverse Wishart probability density function for multiple inputs

Usage

mmNiWpdf(mu, Sigma, U_mu0, U_kappa0, U_nu0, U_lambda0, Log = TRUE)

Arguments

mu
data matrix of dimension \( p \times n \), \( p \) being the dimension of the data and \( n \) the number of data points, where each column is an observed mean vector.
List of length $n$ of observed variance-covariance matrices, each of dimensions $p \times p$.

Mean vectors matrix of dimension $p \times K$, $K$ being the number of distributions for which the density probability has to be evaluated.

Vector of length $K$ of scale parameters.

Vector of length $K$ of degree of freedom parameters.

List of length $K$ of variance-covariance matrices, each of dimensions $p \times p$.

Logical flag for returning the log of the probability density function. Defaults is TRUE.

Value

Matrix of densities of dimension $K \times n$
References

doi:10.1214/18AOAS1209

Description

Probability density function of structured Normal inverse Wishart (sNiW) for multiple inputs, on the log scale.

Usage

mmsNiWlogpdf(U_xi, U_psi, U_Sigma, U_xi0, U_psi0, U_B0, U_Sigma0, U_df0)

Arguments

- **U_xi**
a list of length n of observed mean vectors, each of dimension p
- **U_psi**
a list of length n of observed skew vectors of dimension p
- **U_Sigma**
a list of length n of observed covariance matrices, each of dimension p x p
- **U_xi0**
a list of length K of mean vector parameters for sNiW, each of dimension p
- **U_psi0**
a list of length K of mean vector parameters for sNiW, each of dimension p
- **U_B0**
a list of length K of structuring matrix parameters for sNiW, each of dimension 2 x 2
- **U_Sigma0**
a list of length K of covariance matrix parameters for sNiW, each of dimension p x p
- **U_df0**
a list of length K of degrees of freedom parameters for sNiW, each of dimension p x p

Examples

```r
hyperG0 <- list()
hyperG0$b_xi <- c(-1.6983129, -0.4819131)
hyperG0$b_psi <- c(-0.0641866, -0.7606068)
hyperG0$kappa <- 0.001
hyperG0$D_xi <- 16.951313
hyperG0$D_psi <- 1.255192
hyperG0$nu <- 27.67656
hyperG0$lambda <- matrix(c(2.3397761, -0.3975259, -0.3975259, 1.9601773), ncol=2)

xi_list <- list()
psi_list <- list()
```
```r
S_list <- list() 
for(k in 1:1000){
  NNIW <- rNNiW(hyperG0, diagVar=FALSE)
  xi_list[[k]] <- NNIW["xi"]
  psi_list[[k]] <- NNIW["psi"]
  S_list[[k]] <- NNIW["S"]
}
mmsNiWlogpdf(U_xi=xi_list, U_psi=psi_list, U_Sigma=S_list,
             U_xi0=list(hyperG0$b_xi), U_psi0=list(hyperG0$b_psi),
             U_B0=list(diag(c(hyperG0$D_xi, hyperG0$D_psi))),
             U_Sigma0=list(hyperG0$lambda), U_df0=list(hyperG0$nu))
```

```r
hyperG0 <- list()
hyperG0$b_xi <- c(-1.6983129)
hyperG0$b_psi <- c(-0.0641866)
hyperG0$kappa <- 0.001
hyperG0$D_xi <- 16.951313
hyperG0$D_psi <- 1.255192
hyperG0$nu <- 27.67656
hyperG0$lambda <- matrix(c(2.3397761), ncol=1)
# 'xi_list' <- list()
# 'psi_list' <- list()
S_list <- list()
for(k in 1:1000){
  NNIW <- rNNiW(hyperG0, diagVar=FALSE)
  xi_list[[k]] <- NNIW["xi"]
  psi_list[[k]] <- NNIW["psi"]
  S_list[[k]] <- NNIW["S"]
}
mmsNiWlogpdf(U_xi=xi_list, U_psi=psi_list, U_Sigma=S_list,
             U_xi0=list(hyperG0$b_xi), U_psi0=list(hyperG0$b_psi),
             U_B0=list(diag(c(hyperG0$D_xi, hyperG0$D_psi))),
             U_Sigma0=list(hyperG0$lambda), U_df0=list(hyperG0$nu))
```

---

**mmsNiWpdfC**

* C++ implementation of multivariate structured Normal inverse Wishart probability density function for multiple inputs

### Description

C++ implementation of multivariate structured Normal inverse Wishart probability density function for multiple inputs

### Usage

`mmsNiWpdfC(xi, psi, Sigma, U_xi0, U_psi0, U_B0, U_Sigma0, U_df0, Log = TRUE)`
**Arguments**

- **xi** data matrix of dimensions \( p \times n \) where columns contain the observed mean vectors.
- **psi** data matrix of dimensions \( p \times n \) where columns contain the observed skew parameter vectors.
- **Sigma** list of length \( n \) of observed variance-covariance matrices, each of dimensions \( p \times p \).
- **U_xi0** mean vectors matrix of dimension \( p \times K \), \( K \) being the number of distributions for which the density probability has to be evaluated.
- **U_psi0** skew parameter vectors matrix of dimension \( p \times K \).
- **U_B0** list of length \( K \) of structured scale matrices, each of dimensions \( p \times p \).
- **U_Sigma0** list of length \( K \) of variance-covariance matrices, each of dimensions \( p \times p \).
- **U_df0** vector of length \( K \) of degree of freedom parameters.
- **Log** logical flag for returning the log of the probability density function. Defaults is TRUE.

**Value**

matrix of densities of dimension \( K \times n \)

**References**


---

**mmvnpdfC**

\[ C++ \text{ implementation of multivariate Normal probability density function for multiple inputs} \]

**Description**

C++ implementation of multivariate Normal probability density function for multiple inputs

**Usage**

\[ \text{mmvnpdfC}(x, \text{mean, varcovM, Log = TRUE}) \]
Arguments

- **x**: data matrix of dimension $p \times n$, $p$ being the dimension of the data and $n$ the number of data points.
- **mean**: mean vectors matrix of dimension $p \times K$, $K$ being the number of distributions for which the density probability has to be evaluated.
- **varcovM**: list of length $K$ of variance-covariance matrices, each of dimensions $p \times p$.
- **Log**: logical flag for returning the log of the probability density function. Defaults is TRUE.

Value

matrix of densities of dimension $K \times n$.

Examples

```r
if(require(microbenchmark)){
library(microbenchmark)

microbenchmark(mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE),
               mvnpdfC(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE),
               mmvnpdfC(x=matrix(1.96), mean=matrix(0), varcovM=list(diag(1)), Log=FALSE),
               times=1000L)

microbenchmark(mvnpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1), mean=c(-0.2, 0.3),
                      varcovM=matrix(c(2, 0.2, 0.2, 2), ncol=2), Log=FALSE),
               mvnpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1), mean=c(-0.2, 0.3),
                      varcovM=matrix(c(2, 0.2, 0.2, 2), ncol=2), Log=FALSE),
               mmvnpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
                      mean=matrix(c(-0.2, 0.3), nrow=2, ncol=1),
                      varcovM=list(matrix(c(2, 0.2, 0.2, 2), ncol=2)), Log=FALSE),
               times=1000L)

microbenchmark(mvnpdf(x=matrix(c(rep(1.96,2),rep(0,2)), nrow=2, ncol=2),
                      mean=list(c(0,0),c(-1,-1), c(1.5,1.5)),
                      varcovM=list(diag(2),10*diag(2), 20*diag(2)), Log=FALSE),
               mvnpdfC(x=matrix(c(rep(1.96,2),rep(0,2)), nrow=2, ncol=2),
                      mean=list(c(0,0,-1,-1, 1.5,1.5),
                      varcovM=list(diag(2),10*diag(2), 20*diag(2)), Log=FALSE),
               times=1000L)
}
else{
cat("package 'microbenchmark' not available\n")
}
```

**mmvnpdfC**

C++ implementation of multivariate skew Normal probability density function for multiple inputs

Description

C++ implementation of multivariate skew Normal probability density function for multiple inputs
Usage

\texttt{mmvsnpdfC(x, xi, psi, sigma, Log = TRUE)}

Arguments

- **x**: data matrix of dimension \(p \times n\), \(p\) being the dimension of the data and \(n\) the number of data points.
- **xi**: mean vectors matrix of dimension \(p \times K\), \(K\) being the number of distributions for which the density probability has to be evaluated.
- **psi**: skew parameter vectors matrix of dimension \(p \times K\).
- **sigma**: list of length \(K\) of variance-covariance matrices, each of dimensions \(p \times p\).
- **Log**: logical flag for returning the log of the probability density function. Default is \(TRUE\).

Value

matrix of densities of dimension \(K \times n\).

Author(s)

Boris Hejblum

Examples

\begin{verbatim}
mmvsnpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
           xi=matrix(c(0, 0)), psi=matrix(c(1, 1), ncol=1),
           sigma=list(diag(2)), Log=FALSE)
mmvsnpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
           xi=matrix(c(0, 0)), psi=matrix(c(1, 1), ncol=1),
           sigma=list(diag(2)))
if(require(microbenchmark)){
  library(microbenchmark)
  microbenchmark(mmvsnpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
                         xi=matrix(c(0, 0)), psi=matrix(c(1, 1), ncol=1),
                         sigma=list(diag(2), Log=FALSE),
                         mmvsnpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
                                   xi=matrix(c(0, 0)), psi=matrix(c(1, 1), ncol=1),
                                   sigma=list(diag(2), Log=FALSE),
                                   times=1000L)
  microbenchmark(mmvsnpdf(x=matrix(c(rep(1.96,2),rep(0,2)), nrow=2, ncol=2),
                          xi=matrix(c(0, 0), c(-1,-1), c(1.5,1.5)),
                          sigma=diag(2), 10*diag(2), 20*diag(2)),
  mmvsnpdfC(x=matrix(c(rep(1.96,2),rep(0,2)), nrow=2, ncol=2),
             xi=matrix(c(0, 0), c(-1,-1), c(1.5,1.5), nrow=2, ncol=3),
             psi=matrix(c(0, 0, 1.5, -1, 0.5, -1.5), nrow=2, ncol=3),
             sigma=list(diag(2), 10*diag(2), 20*diag(2), Log=FALSE),
             times=1000L)
}
\end{verbatim}
Description

C++ implementation of multivariate Normal probability density function for multiple inputs

Usage

mmvstpdfC(x, xi, psi, sigma, df, Log = TRUE)

Arguments

- **x**: data matrix of dimension \(p \times n\), \(p\) being the dimension of the data and \(n\) the number of data points.
- **xi**: mean vectors matrix of dimension \(p \times K\), \(K\) being the number of distributions for which the density probability has to be evaluated.
- **psi**: skew parameter vectors matrix of dimension \(p \times K\).
- **sigma**: list of length \(K\) of variance-covariance matrices, each of dimensions \(p \times p\).
- **df**: vector of length \(K\) of degree of freedom parameters.
- **Log**: logical flag for returning the log of the probability density function. Defaults is \text{TRUE}.

Value

matrix of densities of dimension \(K \times n\).

Author(s)

Boris Hejblum

Examples

```r
mmvstpdfC(x = matrix(c(3.399890, -5.936962), ncol=1), xi=matrix(c(0.2528859, -2.4234067)),
psi=matrix(c(11.20536, -12.51052), ncol=1),
sigma=list(matrix(c(0.2134011, -0.0382573, -0.0382573, 0.2660086), ncol=2)),
df=c(7.784106)
)  

mvstpdf(x = matrix(c(3.399890, -5.936962), ncol=1), xi=c(0.2528859, -2.4234067),
psi=c(11.20536, -12.51052),
sigma=matrix(c(0.2134011, -0.0382573, -0.0382573, 0.2660086), ncol=2),
df=c(7.784106)
)```

# skew-normal limit

```r
mmvsnpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=matrix(c(0, 0)), psi=matrix(c(1, 1),ncol=1), sigma=list(diag(2))
)
mvstpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=c(0, 0), psi=c(1, 1), sigma=diag(2),
    df=100000000
)
mmvstpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=matrix(c(0, 0)), psi=matrix(c(1, 1),ncol=1), sigma=list(diag(2)),
    df=100000000
)
```

# non-skewed limit

```r
mmvtpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    mean=matrix(c(0, 0)), varcovM=list(diag(2)),
    df=10
)
mmvstpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=matrix(c(0, 0)), psi=matrix(c(0, 0),ncol=1), sigma=list(diag(2)),
    df=10
)
```

if(require(microbenchmark)){
library(microbenchmark)
microbenchmark(mvstpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=c(0, 0), psi=c(1, 1),
    sigma=diag(2), df=10),
    mmvstpdfC(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=matrix(c(0, 0)), psi=matrix(c(0, 0),ncol=1), sigma=list(diag(2)),
    df=10),
    times=1000L)
} else{
  cat("package \textquotesingle microbenchmark\textquotesingle not available\n")
}
```

---

**mmvtpdfC**

`C++ implementation of multivariate Normal probability density function for multiple inputs**

### Description

C++ implementation of multivariate Normal probability density function for multiple inputs

### Usage

```r
mmvtpdfC(x, mean, varcovM, df, Log = TRUE)
```
**Arguments**

- **x**: data matrix of dimension \( p \times n \), \( p \) being the dimension of the data and \( n \) the number of data points.

- **mean**: mean vectors matrix of dimension \( p \times K \), \( K \) being the number of distributions for which the density probability has to be evaluated.

- **varcovM**: list of length \( K \) of variance-covariance matrices, each of dimensions \( p \times p \).

- **df**: vector of length \( K \) of degree of freedom parameters.

- **Log**: logical flag for returning the log of the probability density function. Defaults is \( \text{TRUE} \).

**Value**

matrix of densities of dimension \( K \times n \).

**Author(s)**

Boris Hejblum

**Examples**

```r
mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE)
mvtpdf(x=matrix(1.96), mean=0, varcovM=diag(1), df=10000000, Log=FALSE)
mmvtpdfC(x=matrix(1.96), mean=matrix(0), varcovM=list(diag(1)), df=10000000, Log=FALSE)

mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1))
mvtpdf(x=matrix(1.96), mean=0, varcovM=diag(1), df=10000000)
mmvtpdfC(x=matrix(1.96), mean=matrix(0), varcovM=list(diag(1)), df=10000000)

mvtpdf(x=matrix(1.96), mean=0, varcovM=diag(1), df=10)
mmvtpdfC(x=matrix(1.96), mean=matrix(0), varcovM=list(diag(1)), df=10)

if(require(microbenchmark)){
  library(microbenchmark)
  microbenchmark(mvtpdf(x=matrix(1.96), mean=0, varcovM=diag(1), df=1, Log=FALSE),
                mmvtpdfC(x=matrix(1.96), mean=matrix(0), varcovM=list(diag(1)),
                         df=c(1), Log=FALSE),
                times=10000L)
} else{
  cat("package 'microbenchmark' not available\n")
}
```
mvnlikC

\[ C++ \text{ implementation of multivariate Normal probability density function for multiple inputs} \]

**Description**

C++ implementation of multivariate Normal probability density function for multiple inputs

**Usage**

\[
\text{mvnlikC}(x, c, \text{clustval}, \mu, \sigma, \text{loglik} = \text{TRUE})
\]

**Arguments**

- **x**: data matrix of dimension p x n, p being the dimension of the data and n the number of data points
- **c**: integer vector of cluster allocations with values from 1 to K
- **clustval**: vector of unique values from c in the order corresponding to the storage of cluster parameters in \( \xi, \psi, \) and \( \text{varcovM} \)
- **mu**: mean vectors matrix of dimension p x K, K being the number of clusters
- **sigma**: list of length K of variance-covariance matrices, each of dimensions p x p.
- **loglik**: logical flag or returning the log-likelihood instead of the likelihood. Default is TRUE.

**Value**

A list:

- "indiv": vector of likelihood of length n;
- "clust": vector of likelihood of length K;
- "total": total (log)-likelihood;

**Author(s)**

Boris Hejblum
**Description**

multivariate-Normal probability density function

**Usage**

`mvnpdf(x, mean, varcovM, Log = TRUE)`

**Arguments**

- `x`: p x n data matrix with n the number of observations and p the number of dimensions
- `mean`: mean vector or list of mean vectors (either a vector, a matrix or a list)
- `varcovM`: variance-covariance matrix or list of variance-covariance matrices (either a matrix or a list)
- `Log`: logical flag for returning the log of the probability density function. Defaults is TRUE.

**Author(s)**

Boris P. Hejblum

**See Also**

`mvnpdf`, `mmvnpdfC`

**Examples**

```r
mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE)
dnorm(1.96)
```

```r
mvnpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
       mean=c(0, 0), varcovM=diag(2), Log=FALSE)
```
mvnpdfC

C++ implementation of multivariate normal probability density function for multiple inputs

Description

Based on the implementation from Nino Hardt and Dicko Ahmadou [https://gallery.rcpp.org/articles/dmvnorm arma/](https://gallery.rcpp.org/articles/dmvnorm_arma/) (accessed in August 2014)

Usage

mvnpdfC(x, mean, varcovM, Log = TRUE)

Arguments

- **x**: data matrix
- **mean**: mean vector
- **varcovM**: variance covariance matrix
- **Log**: logical flag for returning the log of the probability density function. Default is TRUE

Value

vector of densities

Author(s)

Boris P. Hejblum

Examples

mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE)
mvnpdfC(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE)
mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1))
mvnpdfC(x=matrix(1.96), mean=0, varcovM=diag(1))

if(require(microbenchmark)){
  library(microbenchmark)
  microbenchmark(dnorm(1.96),
    mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE),
    mvnpdfC(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE),
    times=10000L)
} else{
  cat("package 'microbenchmark' not available\n")
}
mvsnlikC

C++ implementation of multivariate skew normal likelihood function for multiple inputs

Description
C++ implementation of multivariate skew normal likelihood function for multiple inputs

Usage
mvsnlikC(x, c, clustval, xi, psi, sigma, loglik = TRUE)

Arguments
x  data matrix of dimension p x n, p being the dimension of the data and n the number of data points

clustval  integer vector of cluster allocations with values from 1 to K

xi  vector of unique values from c in the order corresponding to the storage of cluster parameters in xi, psi, and sigma

psi  mean vectors matrix of dimension p x K, K being the number of clusters

sigma  skew parameter vectors matrix of dimension p x K

loglik  list of length K of variance-covariance matrices, each of dimensions p x p.

loglik  logical flag or returning the log-likelihood instead of the likelihood. Default is TRUE.

Value
a list:

"indiv":  vector of likelihood of length n;

"clust":  vector of likelihood of length K;

"total":  total (log)-likelihood;

Author(s)
Boris Hejblum
multivariate Skew-Normal probability density function

**Description**

multivariate Skew-Normal probability density function

**Usage**

```r
mvsnpdf(x, xi, sigma, psi, Log = TRUE)
```

**Arguments**

- `x`  
  p x n data matrix with n the number of observations and p the number of dimensions
- `xi`  
  mean vector or list of mean vectors (either a vector, a matrix or a list)
- `sigma`  
  variance-covariance matrix or list of variance-covariance matrices (either a matrix or a list)
- `psi`  
  skew parameter vector or list of skew parameter vectors (either a vector, a matrix or a list)
- `Log`  
  logical flag for returning the log of the probability density function. Defaults is TRUE.

**See Also**

`mvnpdf`, `mmvsnpdf`

**Examples**

```r
mvsnpdf(x=matrix(1.96), mean=0, varcovM=diag(1), Log=FALSE)
dnorm(1.96)
mvsnpdf(x=matrix(rep(1.96,1), nrow=1, ncol=1),
  xi=c(0), psi=c(0), sigma=diag(1),
  Log=FALSE)
)mvsnpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
  xi=c(0, 0), psi=c(1, 1), sigma=diag(2))
N=50000#00
Yn <- rnorm(n=N, mean=0, sd=1)
Z <- rtruncnorm(n=N, a=0, b=Inf, mean=0, sd=1)
eps <- rnorm(n=N, mean=0, sd=1)
psi <- 10
Ysn <- psi*Z + eps
```
nu <- 1.5
W <- rgamma(n=N, shape=nu/2, rate=nu/2)
Yst = Ysn / sqrt(W)

library(reshape2)
library(ggplot2)
data2plot <- melt(cbind.data.frame(Ysn, Yst))

#pdf(file="ExSNST.pdf", height=5, width=4)
p <- (ggplot(data=data2plot)
  + geom_density(aes(x=value, fill=variable, alpha=variable), col="black"#, lwd=1.1)
  + theme_bw()
  + xlim(-15,100)
  + theme(legend.position="bottom")
  + scale_fill_manual(values=alpha(c("#F8766D", "#00B0F6"),c(0.2,0.45)),
                      name = "",
                      labels=c("Y~SN(0,1,10) ", "Y~ST(0,1,10,1.5)")
  )
  + scale_alpha_manual(guide=FALSE, values=c(0.25, 0.45))
  + xlab("Y")
  + ylim(0,0.08)
  + ylab("Density")
  + guides(fill = guide_legend(override.aes = list(colour = NULL)))
  + theme(legend.key = element_rect(colour = "black"))
)
p
#dev.off()

---

mvstlikC

C++ implementation of multivariate skew t likelihood function for multiple inputs

Description

C++ implementation of multivariate skew t likelihood function for multiple inputs

Usage

mvstlikC(x, c, clustval, xi, psi, sigma, df, loglik = TRUE)

Arguments

x data matrix of dimension p x n, p being the dimension of the data and n the number of data points
c integer vector of cluster allocations with values from 1 to K
clustval vector of unique values from c in the order corresponding to the storage of cluster parameters in xi, psi, and sigma
mvstpdf

xi  mean vectors matrix of dimension p x K, K being the number of clusters
psi skew parameter vectors matrix of dimension p x K
sigma list of length K of variance-covariance matrices, each of dimensions p x p.
df vector of length K of degree of freedom parameters.
loglik logical flag or returning the log-likelihood instead of the likelihood. Default is TRUE.

Value

a list:
"indiv": vector of likelihood of length n;
"clust": vector of likelihood of length K;
"total": total (log)-likelihood;

Author(s)

Boris Hejblum

mvstpdf  multivariate skew-t probability density function

Description

multivariate skew-t probability density function

Usage

mvstpdf(x, xi, sigma, psi, df, Log = TRUE)

Arguments

x  p x n data matrix with n the number of observations and p the number of dimensions
xi  mean vector or list of mean vectors (either a vector, a matrix or a list)
sigma variance-covariance matrix or list of variance-covariance matrices (either a matrix or a list)
psi skew parameter vector or list of skew parameter vectors (either a vector, a matrix or a list)
df  a numeric vector or a list of the degrees of freedom (either a vector or a list)
Log  logical flag for returning the log of the probability density function. Defaults is TRUE.

See Also

mvtpdf, mvsnpdf, mmvstpdfC, mvstlikC
mvtpdf

**Examples**

```r
mvstpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=c(0, 0), psi=c(1, 1), sigma=diag(2),
    df=100000000, Log=FALSE)
mvstpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    xi=c(0, 0), psi=c(1, 1), sigma=diag(2),
    Log=FALSE)
```

---

**mvtpdf**

*multivariate Student's t-distribution probability density function*

**Description**

multivariate Student's t-distribution probability density function

**Usage**

```r
mvtpdf(x, mean, varcovM, df, Log = TRUE)
```

**Arguments**

- **x**: p x n data matrix with n the number of observations and p the number of dimensions
- **mean**: mean vector or list of mean vectors (either a vector, a matrix or a list)
- **varcovM**: variance-covariance matrix or list of variance-covariance matrices (either a matrix or a list)
- **df**: a numeric vector or a list of the degrees of freedom (either a vector or a list)
- **Log**: logical flag for returning the log of the probability density function. Defaults is TRUE.

**Examples**

```r
mvtpdf(x=matrix(1.96), mean=0, varcovM=diag(1), df=100000000)
mvnpdf(x=matrix(1.96), mean=0, varcovM=diag(1))
mvtpdf(x=matrix(1.96), mean=0, varcovM=diag(1), df=10)
```
mvtpdf(x=matrix(rep(1.96,2), nrow=2, ncol=1),
    mean=c(0, 0), varcovM=diag(2), df=10)

---

**NuMatParC**

* C++ implementation of similarity matrix computation using pre-computed distances

### Description

C++ implementation of similarity matrix computation using pre-computed distances

### Usage

```r
NuMatParC(c, d)
```

### Arguments

- `c` an MCMC partitions of length `n`.
- `d` a symmetric `n x n` matrix containing distances between each group distributions.

### Author(s)

Boris Hejblum, Chariff Alkhassim

### Examples

```r
c <- c(1,1,2,3,2,3)
d <- matrix(runif(length(c)^2),length(c))
NuMatParC(c,d)
```

---

**plot_ConvDPM**

* Convergence diagnostic plots

### Description

Convergence diagnostic plots
plot_DPM

Usage

plot_DPM(
  z,
  U_mu = NULL,
  U_Sigma = NULL,
  m, 
  i, 
  alpha = "?",
  U_SS = NULL, 
  dims2plot = 1:nrow(z),
  ellipses = ifelse(length(dims2plot) < 3, TRUE, FALSE),
  gg.add = list(theme()) 
)

Description

Plot of a Dirichlet process mixture of gaussian distribution partition

Usage

plot_DPM( 
    MCMCsample, 
    from = 1, 
    to = length(MCMCsample$logposterior_list),
    shift = 0,
    thin = 1,
    ... 
)

Arguments

MCMCsample a DPMMclust or summaryDPMMclust object.
from the MCMC iteration from which the plot should start. Default is 1.
to the MCMC iteration up until which the plot should stop. Default is 1.
shift a number of initial iterations not to be displayed. Default is 0.
thin integer giving the spacing at which MCMC iterations are kept. Default is 1, i.e. no thinning.
... further arguments passed to or from other methods
**Arguments**

- **z**: data matrix $d \times n$ with $d$ dimensions in rows and $n$ observations in columns.
- **U_mu**: either a list or a matrix containing the current estimates of mean vectors of length $d$ for each cluster. Default is NULL in which case $U_SS$ has to be provided.
- **U_Sigma**: either a list or an array containing the $d \times d$ current estimates for covariance matrix of each cluster. Default is NULL in which case $U_SS$ has to be provided.
- **m**: vector of length $n$ containing the number of observations currently assigned to each clusters.
- **c**: allocation vector of length $n$ indicating which observation belongs to which clusters.
- **i**: current MCMC iteration number.
- **alpha**: current value of the DP concentration parameter.
- **U_SS**: a list containing "mu" and "S". Default is NULL in which case $U_mu$ and $U_Sigma$ have to be provided.
- **dims2plot**: index vector, subset of $1:d$ indicating which dimensions should be drawn. Default is all of them.
- **ellipses**: a logical flag indicating whether ellipses should be drawn around clusters. Default is TRUE if only 2 dimensions are plotted, FALSE otherwise.
- **gg.add**: a list of instructions to add to the ggplot2 instruction (see `gg-add`). Default is `list(theme())`, which adds nothing to the plot.

**Author(s)**

Boris Hejblum

---

**plot_DPMsn**

*Plot of a Dirichlet process mixture of skew normal distribution partition*

**Description**

Plot of a Dirichlet process mixture of skew normal distribution partition

**Usage**

```r
plot_DPMsn(
  z,
  c,
  i = "",
  alpha = "?",
  U_SS,
  dims2plot = 1:nrow(z),
  ellipses = ifelse(length(dims2plot) < 3, TRUE, FALSE),
  gg.add = list(theme()),
  nbsim_dens = 1000
)
```
Arguments

- **z**: data matrix d x n with d dimensions in rows and n observations in columns.
- **c**: allocation vector of length n indicating which observation belongs to which clusters.
- **i**: current MCMC iteration number.
- **alpha**: current value of the DP concentration parameter.
- **U_SS**: a list containing "xi", "psi", "S", and "df".
- **dims2plot**: index vector, subset of 1:d indicating which dimensions should be drawn. Default is all of them.
- **ellipses**: a logical flag indicating whether ellipses should be drawn around clusters. Default is TRUE if only 2 dimensions are plotted, FALSE otherwise.
- **gg.add**: A list of instructions to add to the ggplot2 instruction (see gg-add). Default is list(theme()), which adds nothing to the plot.
- **nbsim_dens**: number of simulated points used for computing clusters density contours in 2D plots. Default is 1000 points.

Author(s)

Boris Hejblum

---

**plot_DPMst**

*Plot of a Dirichlet process mixture of skew t-distribution partition*

Description

Plot of a Dirichlet process mixture of skew t-distribution partition

Usage

```r
plot_DPMst(
  z,
  c,
  i = "",
  alpha = "?",
  U_SS,
  dims2plot = 1:nrow(z),
  ellipses = ifelse(length(dims2plot) < 3, TRUE, FALSE),
  gg.add = list(theme()),
  nbsim_dens = 1000,
  nice = FALSE
)
```
postProcess.DPMMclust

Arguments

- **z**: data matrix \( d \times n \) with \( d \) dimensions in rows and \( n \) observations in columns.
- **c**: allocation vector of length \( n \) indicating which observation belongs to which clusters.
- **i**: current MCMC iteration number.
- **alpha**: current value of the DP concentration parameter.
- **U_SS**: a list containing "xi", "psi", "S", and "df".
- **dims2plot**: index vector, subset of \( 1:d \) indicating which dimensions should be drawn. Default is all of them.
- **ellipses**: a logical flag indicating whether ellipses should be drawn around clusters. Default is TRUE if only 2 dimensions are plotted, FALSE otherwise.
- **gg.add**: A list of instructions to add to the ggplot2 instruction (see gg-add). Default is list(theme()), which adds nothing to the plot.
- **nbsim_dens**: number of simulated points used for computing clusters density contours in 2D plots. Default is 1000 points.
- **nice**: logical flag changing the plot looks. Default is FALSE.

Author(s)

Boris Hejblum

Description

Post-processing Dirichlet Process Mixture Models results to get a mixture distribution of the posterior locations

Usage

```r
postProcess.DPMMclust(
  x,
  burnin = 0,
  thin = 1,
  gs = NULL,
  lossFn = "F-measure",
  K = 10,
  ...
)
```
Arguments

- **x**: a DPMMclust object.
- **burnin**: integer giving the number of MCMC iterations to burn (default is half)
- **thin**: integer giving the spacing at which MCMC iterations are kept. Default is 1, i.e. no thinning.
- **gs**: optional vector of length \( n \) containing the gold standard partition of the \( n \) observations to compare to the point estimate.
- **lossFn**: character string specifying the loss function to be used. Either "F-measure" or "Binder" (see Details). Default is "F-measure".
- **K**: integer giving the number of mixture components. Default is 10.
- **...**: further arguments passed to or from other methods

Details

The cost of a point estimate partition is calculated using either a pairwise coincidence loss function (Binder), or 1-Fmeasure (F-measure).

Value

- **a list**:
  - **burnin**: an integer passing along the burnin argument
  - **thin**: an integer passing along the thin argument
  - **lossFn**: a character string passing along the lossFn argument
  - **point_estim**:
    - **loss**:
    - **index_estim**:

Author(s)

- Boris Hejblum

See Also

- `similarityMat`
- `summary.DPMMclust`
Methods for a summary of a DPMMclust object

Description

Methods for a summary of a DPMMclust object

Usage

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

## S3 method for class 'summaryDPMMclust'
plot(
  x,
  hm = FALSE,
  nbsim_densities = 5000,
  hm_subsample = NULL,
  hm_order_by_clust = TRUE,
  gg.add = list(theme_bw()),
  ...
)
```

Arguments

- `x`: a `summaryDPMMclust` object.
- `...`: further arguments passed to or from other methods.
- `hm`: logical flag to plot the heatmap of the similarity matrix. Default is `FALSE`.
- `nbsim_densities`: the number of simulated observations to be used to plot the density lines of the clusters in the point estimate partition plot.
- `hm_subsample`: an integer designating the number of observations to use when plotting the heatmap. Used only if `hm` is `TRUE`. Default is `NULL` in which no subsampling is done and all observations are plotted.
- `hm_order_by_clust`: a logical flag indicating whether observations should be ordered according to the point estimate first. Used only if `hm` is `TRUE`. Default is `TRUE`.
- `gg.add`: a list of instructions to add to the ggplot2 instruction (see `gg-add`). Default is `list(theme())`, which adds nothing to the plot.

Author(s)

Boris Hejblum
Description

Construction of an Empirical based prior

Usage

priormix(sDPMclust, nu0add = 5)

Arguments

sDPMclust an object of class summary.DPMMclust
nu0add an additional value integer added to hyperprior parameter nu (increase to avoid non positive definite matrix sampling)

See Also

summary.DPMMclust

Examples

rm(list=ls())

# Number of data
n <- 2000
set.seed(123)
#set.seed(4321)

d <- 2
ncl <- 4

# Sample data
sdev <- array(dim=c(d,d,ncl))

xi <- matrix(nrow=d, ncol=ncl, c(-1.5, 1.5, 1.5, 1.5, 2, -2.5, -2.5, -3))
#xi <- matrix(nrow=d, ncol=ncl, c(-0.5, 0, 0.5, 0, 0.5, -1, -1, 1))
psi <- matrix(nrow=d, ncol=4, c(0.4, -0.6, 0.8, 0, 0.3, -0.7, -0.3, -0.8))
nu <- c(100,15,8,5)
p <- c(0.15, 0.05, 0.5, 0.3) # frequence des clusters
sdev[, ,1] <- matrix(nrow=d, ncol=d, c(0.3, 0, 0, 0.3))
sdev[, ,2] <- matrix(nrow=d, ncol=d, c(0.1, 0, 0, 0.3))
sdev[, ,3] <- matrix(nrow=d, ncol=d, c(0.3, 0.15, 0.15, 0.3))
sdev[, ,4] <- .3*diag(2)
c <- rep(0,n)
w <- rep(1,n)
z <- matrix(0, nrow=d, ncol=n)
for(k in 1:n){
c[k] = which(rmultinom(n=1, size=1, prob=p)!=0)
w[k] <- rgamma(1, shape=nu[c[k]]/2, rate=nu[c[k]]/2)
z[,k] <- xi[, c[k]] + psi[, c[k]]*rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=1/sqrt(w[k])) +
         (sdev[, , c[k]]/sqrt(w[k]))%*%matrix(rnorm(d, mean = 0, sd = 1), nrow=d, ncol=1)
#cat(k, "/", n, " observations simulated\n", sep="")
}

# Set parameters of G0
hyperG0 <- list()
hyperG0["b_xi"] <- rowMeans(z)
hyperG0["b_psi"] <- rep(0,d)
hyperG0["kappa"] <- 0.001
hyperG0["D_xi"] <- 100
hyperG0["D_psi"] <- 100
hyperG0["nu"] <- d+1
hyperG0["lambda"] <- diag(apply(z,MARGIN=1, FUN=var))/3

# hyperprior on the Scale parameter of DPM
a <- 0.0001
b <- 0.0001

nbclust_init <- 30

if(interactive()){
    MCMCsample_st <- DPMGibbsSkewT(z, hyperG0, a, b, N=2000, doPlot=FALSE, nbclust_init, diagVar=FALSE)
s <- summary(MCMCsample_st, burnin = 1500, thin=5, posterior_approx=TRUE)
    pmix <- priormix(s)
}

rCRP             Generating cluster data from the Chinese Restaurant Process

Description
Generating cluster data from the Chinese Restaurant Process

Usage
rCRP(n = 1000, alpha = 2, hyperG0, verbose = TRUE)

Arguments
n  number of observations
alpha  concentration parameter
hyperG0  base distribution hyperparameter
verbose  logical flag indicating whether info is written in the console.

Examples

```r
rm(list=ls())

d=2
hyperG0 <- list()
hyperG0["NNiW"] <- list()
hyperG0["NNiW"]["b_xi"] <- rep(0,d)
hyperG0["NNiW"]["b_psi"] <- rep(0,d)
hyperG0["NNiW"]["D_xi"] <- 100
hyperG0["NNiW"]["D_psi"] <- 8
hyperG0["NNiW"]["nu"] <- d+1
hyperG0["NNiW"]["lambda"] <- diag(c(1,1))

hyperG0["scale"] <- list()

set.seed(4321)
N <- 200
alph <- runif(n=1,0.2,2)
GvHD_sims <- rCRP(n=2*N, alpha=alph, hyperG0=hyperG0)
library(ggplot2)
q <- (ggplot(data=cbind.data.frame("D1"=GvHD_sims$data[1,],
"D2"=GvHD_sims$data[2,],
"Cluster"=GvHD_sims$cluster),
  aes(x=D1, y=D2))
  + geom_point(aes(colour=Cluster), alpha=0.6)
  + theme_bw() )

q

if(interactive()){
  MCMCy1 <- DPMGibbsSkewT(z=GvHD_sims$data[,1:N],
    hyperG0$NNiW, a=0.0001, b=0.0001, N=5000,
    doPlot=TRUE, nbclust_init=64, plotevery=500,
    gg.add=list(theme_bw()), diagVar=FALSE)
  s1 <- summary(MCMCy1, burnin=4000, thin=5,
    posterior_approx=TRUE)
  F1 <- FmeasureC(ref=GvHD_sims$cluster[1:N], pred=s1$point_estim$c_est)

  # s <- summary(MCMCy1, burnin=4000, thin=5,
  #   posterior_approx=TRUE, K=1)
  # s2 <- summary(MCMCy1, burnin=4000, thin=5,
  #   posterior_approx=TRUE, K=2)
  # MCMCy2_seqPost<- DPMGibbsSkewT(z=GvHD_sims$data[, (N+1):2*N],
  #   hyperG0=s1$param_post$parameters,
  #   a=s1$param_post$alpha_param$shape,
  #   b=s1$param_post$alpha_param$rate,
```
```r
MCMCy2_seqPost <- DPMGibbsSkewT_SeqPrior(z=GvHD_sims$data[, (N+1):(2*N)],
    prior=s1$param_post, hyperG0=hyperG0$NNiW, , N=1000,
    doPlot=TRUE, nbclust_init=10, plotevery=100,
    gg.add=list(theme_bw()), diagVar=FALSE)

s2_seqPost <- summary(MCMCy2_seqPost, burnin=600, thin=2)
F2_seqPost <- FmeasureC(ref=GvHD_sims$cluster[(N+1):(2*N)], pred=s2_seqPost$point_estim$c_est)

MCMCy2 <- DPMGibbsSkewT(z=GvHD_sims$data[, (N+1):(2*N)],
    hyperG0$NNiW, a=0.0001, b=0.0001, N=5000,
    doPlot=TRUE, nbclust_init=64, plotevery=500,
    gg.add=list(theme_bw()), diagVar=FALSE)
s2 <- summary(MCMCy2, burnin=4000, thin=5)
F2 <- FmeasureC(ref=GvHD_sims$cluster[(N+1):(2*N)], pred=s2$point_estim$c_est)

MCMCtot <- DPMGibbsSkewT(z=GvHD_sims$data,
    hyperG0$NNiW, a=0.0001, b=0.0001, N=5000,
    doPlot=TRUE, nbclust_init=10, plotevery=500,
    gg.add=list(theme_bw()), diagVar=FALSE)
stot <- summary(MCMCtot, burnin=4000, thin=5)
F2tot <- FmeasureC(ref=GvHD_sims$cluster[(N+1):(2*N)], pred=stot$point_estim$c_est[(N+1):(2*N)])

c(F1, F2, F2_seqPost, F2tot)
```

---

### Sampler for the concentration parameter of a Dirichlet process

#### Description

Sampler updating the concentration parameter of a Dirichlet process given the number of observations and a Gamma(a, b) prior, following the augmentation strategy of West, and of Escobar and West.

#### Usage

```r
sample_alpha(alpha_old, n, K, a = 1e-04, b = 1e-04)
```

#### Arguments

- `alpha_old` : the current value of alpha
- `n` : the number of data points
- `K` : current number of cluster
- `a` : shape hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001.
scale hyperparameter of the Gamma prior on the concentration parameter of the Dirichlet Process. Default is 0.0001. If 0 then the concentration is fixed and this function returns a.

Details

A Gamma prior is used.

References


Examples

```r
#Test with a fixed K
alphas <- numeric(N)
for (i in 1:N){
  alphas[i] <- sample_alpha(alpha_old = alphas[i-1], n=n, K=K, a=a, b=b)
}
postalphas <- alphas[floor(N/2):N]
alphaMMSE <- mean(postalphas)
alphaMAP <- density(postalphas)$x[which.max(density(postalphas)$y)]
expK <- sum(alphaMMSE/(alphaMMSE+0:(n-1))
round(expK)
```

```r
prioralpha <- data.frame("alpha"=rgamma(n=5000, a,1/b),
  "distribution" =factor(rep("prior",5000),
  levels=c("prior", "posterior"))
library(ggplot2)
p <- (ggplot(prioralpha, aes(x=alpha))
  + geom_histogram(aes(y=..density..),
    colour="black", fill="white")
  + geom_density(alpha=.2, fill="red")
  + ggtitle(paste("Prior distribution on alpha: Gamma("", a, ",", b, ")\n", sep="")))
```
similarityMat

Computes the co-clustering (or similarity) matrix

Description

Computes the co-clustering (or similarity) matrix

Usage

similarityMat(c, step = 1)

Arguments

c a list of vector of length n. c[[j]][i] is the cluster allocation of observation
i=1...n at iteration j=1...N.

step provide co-clustering every step iterations. Default is 1.

Value

A matrix of size n x n whose term [i, j] is the proportion of MCMC iterations where observation
i and observations j are allocated to the same cluster.

Author(s)

Boris Hejblum
**similarityMatC**  
*C++ implementation*

**Description**

C++ implementation

**Usage**

`similarityMatC(cc)`

**Arguments**

- `cc`  
  a matrix whose columns each represents a (MCMC) partition

**Examples**

```r
c <- list(c(1,1,2,3,2,3), c(1,1,1,2,3,3),c(2,2,1,1,1,1))
similarityMatC(sapply(c, "["))

c2 <- list()
for(i in 1:10){
  c2 <- c(c2, list(rmultinom(n=1, size=200, prob=rexp(n=200))))
}  
similarityMatC(sapply(c2, "["))
```

---

**similarityMat_nocostC**  
*C++ implementation*

**Description**

C++ implementation

**Usage**

`similarityMat_nocostC(cc)`

**Arguments**

- `cc`  
  a matrix whose columns each represents a (MCMC) partition
Examples

c <- list(c(1,1,2,3,2,3), c(1,1,1,2,3,3),c(2,2,1,1,1,1))
similarityMat_nocostC(sapply(c, "["))

c2 <- list()
for(i in 1:10){
  c2 <- c(c2, list(rmultinom(n=1, size=1000, prob=rexp(n=1000))))
}
c3 <- sapply(c2, "[")

if(require(microbenchmark)){
  library(microbenchmark)
  microbenchmark(similarityMat(c3), similarityMat_nocostC(c3), times=2L)
}else{
  cat("package 'microbenchmark' not available\n")
}

summary.DPMMclust

Summarizing Dirichlet Process Mixture Models

Description

Summary methods for DPMMclust objects.

Usage

## S3 method for class 'DPMMclust'
summary(
  object,
  burnin = 0,
  thin = 1,
  gs = NULL,
  lossFn = "Binder",
  posterior_approx = FALSE,
  ...
)

Arguments

object a DPMMclust object.
burnin integer giving the number of MCMC iterations to burn (defaults is half)
thin integer giving the spacing at which MCMC iterations are kept. Default is 1, i.e. no thinning.

... optional vector of length n containing the gold standard partition of the n observations to compare to the point estimate
lossFn character string specifying the loss function to be used. Either "F-measure" or "Binder" (see Details). Default is "Binder".

posterior_approx logical flag whether a parametric approximation of the posterior should be computed. Default is FALSE

... further arguments passed to or from other methods

Details

The cost of a point estimate partition is calculated using either a pairwise coincidence loss function (Binder), or 1-Fmeasure (F-measure).

The number of retained sampled partitions is \( m = (N - \text{burnin}) / \text{thin} \)

Value

a list containing the following elements:

nb_mcmc: an integer giving the value of \( m \), the number of retained sampled partitions, i.e. \((N - \text{burnin}) / \text{thin}\)

burnin: an integer passing along the burnin argument

thin: an integer passing along the thin argument

lossFn: a character string passing along the lossFn argument

clust_distrib: a character string passing along the clust_distrib argument

point_estim: a list containing:

- `c_est`: a vector of length \( n \) containing the point estimated clustering for each observation
- `cost`: a vector of length \( m \) containing the cost of each sampled partition
- `Fmeas`: if `lossFn` is not 'Binder', the \( m \times m \) matrix of total F-measures for each pair of sampled partitions
- `opt_ind`: the index of the point estimate partition among the \( m \) sampled partitions

loss: the loss for the point estimate. NA if `lossFn` is not 'Binder'

param_posterior: a list containing the parametric approximation of the posterior, suitable to be plugged in as prior for a new MCMC algorithm run

mcmc_partitions: a list containing the \( m \) sampled partitions

alpha: a vector of length \( m \) with the values of the alpha DP parameter

index_estim: the index of the point estimate partition among the \( m \) sampled partitions

hyperG0: a list passing along the prior, i.e. the hyperG0 argument

logposterior_list: a list of length \( m \) containing the logposterior and its decomposition, for each sampled partition

U_SS_list: a list of length \( m \) containing the containing the lists of sufficient statistics for all the mixture components, for each sampled partition

data: a \( d \times n \) matrix containing the clustered data

Author(s)

Boris Hejblum
See Also

similarityMat similarityMatC

Description

C++ implementation

Usage

vclust2mcoclustC(c)

Arguments

c is an MCMC partition

Author(s)

Chariff Alkhassim

Examples

cc <- c(1,1,2,3,2,3)
vclust2mcoclustC(cc)
Index

burn.DPMMclust, 4
cluster_est_binder, 4, 54
cluster_est_Fmeasure, 5
cluster_est_Mbinder_norm, 6
cluster_est_pear, 7
connections, 15, 30, 38, 45
cytoScatter, 8

DPMGibbsN, 10, 16, 19, 49
DPMGibbsN_parallel, 14, 49
DPMGibbsN_SeqPrior, 17, 49
DPMGibbsSkewN, 21, 49
DPMGibbsSkewN_parallel, 29, 49
DPMGibbsSkewT, 32, 49
DPMGibbsSkewT_parallel, 37, 49
DPMGibbsSkewT_SeqPrior, 41, 49
DPMGibbsSkewT_SeqPrior_parallel, 44, 49
DPMpost, 3, 48
evalClustLoss, 53
Flimited, 54
Fmeasure_costC, 56
FmeasureC, 55
FmeasureC_no0, 55

lgamma_mv, 57
makeCluster, 15, 29, 37, 45, 49
MAP_sNiW_mmEM, 57
MAP_sNiW_mmEM_vague (MAP_sNiW_mmEM), 57
MAP_sNiW_mmEM_weighted (MAP_sNiW_mmEM), 57
MLE_gamma, 60
MLE_NiW_mmEM, 61
MLE_sNiW, 62
MLE_sNiW_mmEM, 63
mmsNiWpdf, 65
mmsNiWpdfC, 66
mmsNiWlogpdf, 67
mmvsnpdfC, 69, 76
mmvtpdfC, 73
mvlnlikC, 75
mvpdf, 76, 76, 79
mvpdfC, 77
mvslnlikC, 78
mvsnpdf, 79, 81
mvstlikC, 80, 81
mvstpdf, 81
mvpdf, 81, 82

NPflow (NPflow-package), 3
NPflow-package, 3
NuMatParC, 83

plot.summaryDPMMclust
(print.summaryDPMMclust), 89
plot_CovDPN, 83
plot_DPM, 11, 15, 18, 30, 42, 45, 49, 84
plot_DPMsn, 22, 85
plot_DPMst, 34, 38, 86
postProcess.DPMMclust, 19, 87
print.summaryDPMMclust, 89
prioMix, 90

rCRP, 91

sample_alpha, 93
similarityMat, 5–8, 54, 88, 95, 99
similarityMat_nocostC, 7, 96
similarityMatC, 5, 7, 8, 96, 99
summary.DPMMclust, 4, 50, 88, 97
summaryDPMMclust
(print.summaryDPMMclust), 89

vclust2mcoclustC, 99