Package 'COMIX'

January 20, 2025

Type Package

Title Coarsened Mixtures of Hierarchical Skew Kernels

Version 1.0.0

Description Bayesian fit of a Dirichlet Process Mixture with hierarchical multivariate skew normal kernels and coarsened posteriors. For more information, see Gorsky, Chan and Ma (2020) <arXiv:2001.06451>.

License CC0

Encoding UTF-8

Imports Rcpp (>= 0.12.18), dplyr, ggplot2, stringr, coda, tidyr, rlang, methods

Suggests sn, R.rsp

LinkingTo Rcpp, RcppArmadillo, RcppEigen, RcppNumerical

RoxygenNote 7.2.1

VignetteBuilder R.rsp

NeedsCompilation yes

Author S. Gorsky [aut, cre],
C. Chan [ctb],
L. Ma [ctb]

Maintainer S. Gorsky <sgorsky@umass.edu>

Repository CRAN

Date/Publication 2022-11-23 16:20:02 UTC

Contents

`acfParams` *The function computes (and by default plots) estimates of the autocovariance or autocorrelation function for the different parameters of the model. This is a wrapper for coda::acf.*

Description

The function computes (and by default plots) estimates of the autocovariance or autocorrelation function for the different parameters of the model. This is a wrapper for coda::acf.

Usage

```
acfParams(
  res,
  params = c("w", "xi", "xi0", "psi", "G", "E", "eta"),
  only_non_trivial_clusters = TRUE,
  lag.max = NULL,
  type = c("correlation", "covariance", "partial"),
  plot = TRUE,
  ...
)
```

Arguments

`res`	An object of class `COMIX` or `tidyChainCOMIX`.
`params`	A character vector naming the parameters to compute and plot the autocorrelation plots for.
`only_non_trivial_clusters`	Logical, if `TRUE` only compute and/or plot the autocorrelation for the clusters that are estimated to be non-empty.
`lag.max`	maximum lag at which to calculate the autocorrelation. See more details at ?acf.
`type`	Character string giving the type of autocorrelation to be computed. See more details at ?acf.
`plot`	Logical. If `TRUE` (the default) the autocorrelation is plotted.
`...`	Other arguments passed to `acf`.

Value

An `acfParamsCOMIX` object which is a named list, with a named element for each requested parameter. Each element is an object of class `acf` (from the `coda` package). #' @examples library(COMIX) # Number of observations for each sample (row) and cluster (column): njk <- matrix(c(150, 300, 250, 200), nrow = 2, byrow = TRUE)

Dimension of data: p <- 3

Scale and skew parameters for first cluster: Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p) alpha1 <- rep(0, p) alpha1[1] <- -5 # location parameter for first cluster in first sample: xi11 <- rep(0, p) # location parameter for first cluster in second sample (aligned with first): xi21 <- rep(0, p)

Scale and skew parameters for second cluster: Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p) alpha2 <- rep(0, p) alpha2[2] <- 5 # location parameter for second cluster in first sample: xi12 <- rep(3, p) # location parameter for second cluster in second sample (misaligned with first): xi22 <- rep(4, p)

Sample data: set.seed(1) Y <- rbind(sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1), sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2), sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1), sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2))

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10) pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation # Fit the model: res <- comix(Y, C, pmc = pmc, prior = prior)

Relabel to resolve potential label switching issues: res_relab <- relabelChain(res) effssz <- effectiveSampleSize(res_relab, "w") # Or: tidy_chain <- tidyChain(res_relab, "w") acf_w <- acfParams(tidy_chain, "w")

(see vignette for a more detailed example)

`calibrate` *This function aligns multiple samples so that their location parameters are equal.*

Description

This function aligns multiple samples so that their location parameters are equal.

Usage

```
calibrate(x, reference.group = NULL)
```

Arguments

x	An object of class COMIX.
reference.group	An integer between 1 and the number of groups in the data (length(unique(C))). Defaults to NULL. If NULL, the samples are aligned so that their location parameters are set to be at the estimated group location parameter. If an integer, the samples are aligned so that their location parameters are the same as the location parameter of sample reference.group.

Value

A named list of 3:

- Y_cal: a nrow(x$data$Y) $\times$ ncol(x$data$Y) matrix, a calibrated version of the original data.
- calibration_distribution: an xpmcnsave $\times$ ncol(x$data$Y) $\times$ nrow(x$data$Y) array storing the difference between the estimated sample-specific location parameter and the group location parameter for each saved step of the chain.
- calibration_median: a nrow(x$data$Y) $\times$ ncol(x$data$Y) matrix storing the median difference between the estimated sample-specific location parameter and the group location parameter for each saved step of the chain. This matrix is equal to the difference between the uncalibrated data (x$data$Y) and the calibrated data (Y_cal).

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
```

```
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)

# Generate calibrated data:
cal <- calibrateNoDist(res_relab)

# Compare raw and calibrated data: (see plot in vignette)
# par(mfrow=c(1, 2))
# plot(Y, col = C, xlim = range(Y[,1]), ylim = range(Y[,2]) )

# Get posterior estimates for the model parameters:
res_summary <- summarizeChain(res_relab)
# Check for instance, the cluster assignment labels:
table(res_summary$t)
# Indeed the same as
colSums(njk)

# Or examine the skewness parameter for the non-trivial clusters:
res_summary$alpha[ , unique(res_summary$t)]
# And compare those to
cbind(alpha1, alpha2)

# (see vignette for a more detailed example)
```

`calibrateNoDist` *This function aligns multiple samples so that their location parameters are equal.*

Description

This function aligns multiple samples so that their location parameters are equal.

Usage

```
calibrateNoDist(x, reference.group = NULL)
```

Arguments

`x`	An object of class COMIX.
`reference.group`	An integer between 1 and the number of groups in the data (`length(unique(C))`). Defaults to `NULL`. If `NULL`, the samples are aligned so that their location parameters are set to be at the estimated group location parameter. If an integer, the samples are aligned so that their location parameters are the same as the location parameter of sample `reference.group`.

Value

A named list of 2:

- `Y_cal`: a `nrow(x$data$Y)` $\times$ `ncol(x$data$Y)` matrix, a calibrated version of the original data.
- `calibration_median`: a `nrow(x$data$Y)` $\times$ `ncol(x$data$Y)` matrix storing the median difference between the estimated sample-specific location parameter and the group location parameter for each saved step of the chain. This matrix is equal to the difference between the uncalibrated data (`x$data$Y`) and the calibrated data (`Y_cal`).

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3
```

```
# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)

# Generate calibrated data:
cal <- calibrateNoDist(res_relab)

# Compare raw and calibrated data: (see plot in vignette)
# par(mfrow=c(1, 2))
# plot(Y, col = C, xlim = range(Y[,1]), ylim = range(Y[,2]) )

# Get posterior estimates for the model parameters:
res_summary <- summarizeChain(res_relab)
# Check for instance, the cluster assignment labels:
table(res_summary$t)
# Indeed the same as
colSums(njk)
```

```
# Or examine the skewness parameter for the non-trivial clusters:
res_summary$alpha[ , unique(res_summary$t)]
# And compare those to
cbind(alpha1, alpha2)

# (see vignette for a more detailed example)
```

`comix` *This function generates a sample from the posterior of COMIX.*

Description

This function generates a sample from the posterior of COMIX.

Usage

```
comix(Y, C, prior = NULL, pmc = NULL, state = NULL, ncores = 2)
```

Arguments

`Y`	Matrix of the data. Each row represents an observation.
`C`	Vector of the group label of each observation. Labels must be integers starting from 1.
`prior`	A list giving the prior information. If unspecified, a default prior is used. The list includes the following parameters:

- `zeta`: Coarsening parameter. A number between 0 and 1. `zeta` = 1: sample from standard posterior; `zeta` < 1: sample from power posterior. The lower `zeta` is, the more flexible the kernels become.
- `K`: Maximal number of mixture components.
- `eta_prior` Parameters for gamma prior for concentration parameter of the stick breaking process prior for the weights.
- `m0`: Number of degrees of freedom for the inverse Wishart prior for Sigma, the covariance matrix of the kernels.
- `Lambda`: Mean parameter for the inverse Wishart prior for Sigma, the covariance matrix of the kernels.
- `b0`: Mean parameter for the multivariate normal distribution that is the prior for the group mean parameter xi0.
- `B0`: Covariance parameter for the multivariate normal distribution that is the prior for the group mean parameter xi0.
- `e0`: Number of degrees of freedom for the inverse Wishart prior for E_k, the covariance matrix of the multivariate normal from which $\xi_{j,k}$ are drawn.
- `E0`: Mean parameter for the inverse Wishart prior for E_k, the covariance matrix of the multivariate normal from which $\xi_{j,k}$ are drawn.
- `merge_step`: Introduce step to merge mixture components with small KL divergence. Default is `merge_step` = `TRUE`.

- merge_par: Parameter controlling merging radius. Default is merge_par = 0.1.

pmc — A list giving the Population Monte Carlo (PMC) parameters:

- npart: Number of PMC particles.
- nburn: Number of burn-in steps
- nsave: Number of steps in the chain after burn-in.
- nskip: Thinning parameter, number of steps to skip between saving steps after burn-in.
- ndisplay: Display status of chain after every ndisplay steps.

state — A list giving the initial cluster labels:

- t: An integer vector, same length as the number of rows of Y, with cluster labels between 1 and K.

ncores — The number of CPU cores to utilize in parallel. Defaults to 2.

Value

An object of class COMIX, a list of 4:

chain, a named list:

- t: an nsave × nrow(Y) matrix with estimated cluster labels for each saved step of the chain and each observation in the data Y.
- z: a nsave × nrow(Y) matrix with estimated values of the latent $z_{i,j}$ variable for the parameterization of the multivariate skew normal distribution used in the sampler for each saved step of the chain and each observation in the data Y.
- W: an length(unique(C)) × K ×
- nsave: array storing the estimated sample- and cluster-specific weights for each saved step of the chain.
- xi: an length(unique(C)) × (ncol(Y) x K) × nsave array storing the estimated sample- and cluster-specific multivariate skew normal location parameters of the kernel for each saved step of the chain.
- xi0: an ncol(Y) × K ×
- nsave: array storing the estimated cluster-specific group location parameters for each saved step of the chain.
- psi: an ncol(Y) × K × nsave array storing the estimated cluster-specific skew parameters of the kernels in the parameterization of the multivariate skew normal distribution used in the sampler for each saved step of the chain.
- G: an ncol(Y) × (ncol(Y) x K) × nsave array storing the estimated cluster-specific multivariate skew normal scale matrix (in row format) of the kernel used in the sampler for each saved step of the chain.
- E: an ncol(Y) × (ncol(Y) x K) × nsave array storing the estimated covariance matrix (in row format) of the multivariate normal distribution from which the sample- and cluster-specific location parameters are drawn for each saved step of the chain.
- eta: a nsave × 1 matrix storing the estimated Dirichlet Process Mixture concentration parameter for each saved step of the chain.

  - Sigma: an ncol(Y) $\times$ (ncol(Y) x K) $\times$ nsave array storing the estimated cluster-specific multivariate skew normal scale matrix (in row format) of the kernel for each saved step of the chain.
  - alpha: an ncol(Y) $\times$ K $\times$ nsave array storing the estimated cluster-specific skew parameters of the kernel's multivariate skew normal distribution for each saved step of the chain.

- data, a named list that includes the matrix of the data Y and C the vector of the group label of each observation.
- prior and pmc, the lists, as above, that were provided as inputs to the function.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )


# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
```

```
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)

# Generate calibrated data:
cal <- calibrateNoDist(res_relab)

# Compare raw and calibrated data: (see plot in vignette)
# par(mfrow=c(1, 2))
# plot(Y, col = C, xlim = range(Y[,1]), ylim = range(Y[,2]) )

# Get posterior estimates for the model parameters:
res_summary <- summarizeChain(res_relab)
# Check for instance, the cluster assignment labels:
table(res_summary$t)
# Indeed the same as
colSums(njk)

# Or examine the skewness parameter for the non-trivial clusters:
res_summary$alpha[ , unique(res_summary$t)]
# And compare those to
cbind(alpha1, alpha2)

# (see vignette for a more detailed example)
```

`effectiveSampleSize` *This function creates an object that summarizes the effective sample size for the parameters of the model.*

Description

This function creates an object that summarizes the effective sample size for the parameters of the model.

Usage

```
effectiveSampleSize(res, params = c("w", "xi", "xi0", "psi", "G", "E", "eta"))
```

Arguments

res	An object of class COMIX or tidyChainCOMIX.
params	A character vector naming the parameters to compute the effective sample size for.

Value

An effectiveSampleSizeCOMIX object which is a named list, with a named element for each requested parameter. Each element is a data frame that includes the effective sample size for the parameter.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
```

```
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
effssz <- effectiveSampleSize(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
effssz <- effectiveSampleSize(tidy_chain, "w")
# (see vignette for a more detailed example)
```

`gewekeParams` *This function creates an object that summarizes the Geweke convergence diagnostic.*

Description

This function creates an object that summarizes the Geweke convergence diagnostic.

Usage

```
gewekeParams(
  res,
  params = c("w", "xi", "xi0", "psi", "G", "E", "eta"),
  frac1 = 0.1,
  frac2 = 0.5,
  probs = c(0.025, 0.975)
)
```

Arguments

`res`	An object of class `COMIX` or `tidyChainCOMIX`.
`params`	A character vector naming the parameters to compute the Geweke diagnostic for.
`frac1`	Double, fraction to use from beginning of chain.
`frac2`	Double, fraction to use from end of chain.
`probs`	A vector of 2 doubles, probabilities denoting the limits of a confidence interval for the convergence diagnostic.

Value

An gewekeParamsCOMIX object which is a named list, with a named element for each requested parameter. Each element is a data frame that includes the Geweke diagnostic and result of a stationarity test for the parameter.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))
```

```
prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
effssz <- effectiveSampleSize(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
gwk <- gewekeParams(tidy_chain, "w")
# (see vignette for a more detailed example)
```

`heidelParams` *This function creates an object that summarizes the Heidelberg-Welch convergence diagnostic.*

Description

This function creates an object that summarizes the Heidelberg-Welch convergence diagnostic.

Usage

```
heidelParams(
  res,
  params = c("w", "xi", "xi0", "psi", "G", "E", "eta"),
  eps = 0.1,
  pvalue = 0.05
)
```

Arguments

`res`	An object of class `COMIX` or `tidyChainCOMIX`.
`params`	A character vector naming the parameters to compute the Heidelberg-Welch diagnostic for.
`eps`	Target value for ratio of halfwidth to sample mean.
`pvalue`	Significance level to use.

Value

An `heidelParamsCOMIX` object which is a named list, with a named element for each requested parameter. Each element is a data frame that includes the Heidelberg-Welch diagnostic and results of a stationarity test for the parameter.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)
```

```
# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
effssz <- effectiveSampleSize(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
hd <- heidelParams(tidy_chain, "w")
# (see vignette for a more detailed example)
```

`plotEffectiveSampleSize` *This function creates plots for the effective sample size for the parameters of the model.*

Description

This function creates plots for the effective sample size for the parameters of the model.

Usage

```
plotEffectiveSampleSize(effssz, param)
```

Arguments

`effssz`	An object of class `effectiveSampleSizeCOMIX` as created by the function `effectiveSampleSize`.
`param`	Character, naming the parameter to create a plot of effective sample sizes.

Value

A `ggplot2` plot containing the effective sample size plot.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
```

```
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
effssz <- effectiveSampleSize(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
effssz <- effectiveSampleSize(tidy_chain, "w")
plotEffectiveSampleSize(effssz, "w")
# (see vignette for a more detailed example)
```

`plotGewekeParams` *This function creates plots for the Geweke diagnostic and results of test of stationarity for the parameters of the model.*

Description

This function creates plots for the Geweke diagnostic and results of test of stationarity for the parameters of the model.

Usage

```
plotGewekeParams(gwk, param)
```

Arguments

gwk	An object of class gewekeParamsCOMIX as created by the function gewekeParams.
param	Character, naming the parameter to create a plot of the Geweke diagnostic for.

Value

A ggplot2 plot containing the Geweke diagnostic plot.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
```

```
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
effssz <- effectiveSampleSize(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
gwk <- gewekeParams(tidy_chain, "w")
plotGewekeParams(gwk, "w")
# (see vignette for a more detailed example)
```

`plotHeidelParams` *This function creates plots for the Heidelberg-Welch diagnostic and results of test of stationarity for the parameters of the model.*

Description

This function creates plots for the Heidelberg-Welch diagnostic and results of test of stationarity for the parameters of the model.

Usage

```
plotHeidelParams(hd, param)
```

Arguments

`hd`	An object of class `heidelParamsCOMIX` as created by the function `heidelParams`.
`param`	Character, naming the parameter to create a plot of the Heidelberg-Welch diagnostic for.

Value

A `ggplot2` plot containing the Heidelberg-Welch diagnostic plot.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)
```

```
# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
effssz <- effectiveSampleSize(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
hd <- heidelParams(tidy_chain, "w")
plotHeidelParams(hd, "w")
# (see vignette for a more detailed example)
```

`plotTracePlots` *This function creates trace plots for different parameters of the MCMC chain.*

Description

This function creates trace plots for different parameters of the MCMC chain.

Usage

```
plotTracePlots(res, param)
```

Arguments

`res`	An object of class `COMIX` or `tidyChainCOMIX`.
`param`	Character, naming the parameter to create a trace plot for.

Value

A `ggplot2` plot containing the trace plot.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
```

```
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
plotTracePlots(res_relab, "w")
# Or:
tidy_chain <- tidyChain(res_relab, "w")
plotTracePlots(tidy_chain, "w")
# (see vignette for a more detailed example)
```

`relabelChain` *This function relabels the chain to avoid label switching issues.*

Description

This function relabels the chain to avoid label switching issues.

Usage

```
relabelChain(res)
```

Arguments

res	An object of class COMIX.

Value

An object of class COMIX where `res$chain$t` is replaced with the new labels.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )

# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
```

```
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)

# Generate calibrated data:
cal <- calibrateNoDist(res_relab)

# Compare raw and calibrated data: (see plot in vignette)
# par(mfrow=c(1, 2))
# plot(Y, col = C, xlim = range(Y[,1]), ylim = range(Y[,2]) )

# Get posterior estimates for the model parameters:
res_summary <- summarizeChain(res_relab)
# Check for instance, the cluster assignment labels:
table(res_summary$t)
# Indeed the same as
colSums(njk)

# Or examine the skewness parameter for the non-trivial clusters:
res_summary$alpha[ , unique(res_summary$t)]
# And compare those to
cbind(alpha1, alpha2)

# (see vignette for a more detailed example)
```

`summarizeChain` *This function provides post-hoc estimates of the model parameters.*

Description

This function provides post-hoc estimates of the model parameters.

Usage

```
summarizeChain(res)
```

Arguments

`res`	An object of class COMIX.

Value

A named list:

- `xi0`: a `ncol(res$data$Y)` $\times$ `res$prior$K` matrix storing the posterior mean of the group location parameter.
- `psi`: a `ncol(res$data$Y)` $\times$ `res$prior$K` matrix storing the posterior mean of the multivariate skew normal kernels skewness parameter (in the parameterization used in the sampler).
- `alpha`: a `ncol(res$data$Y)` $\times$ `res$prior$K` matrix storing the posterior mean of the multivariate skew normal kernels skewness parameter.
- `W`: a `length(unique(res$data$C))` $\times$ `res$prior$K` matrix storing the posterior mean of the mixture weights for each sample and cluster.
- `xi`: an `length(unique(res$data$C))` $\times$ `ncol(res$data$Y)` $\times$ `res$prior$K` array storing the the posterior mean of the multivariate skew normal kernels location parameter for each sample and cluster.
- `Sigma`: an `ncol(res$data$Y)` $\times$ `ncol(res$data$Y)` $\times$ `res$prior$K` array storing the the posterior mean of the scaling matrix of the multivariate skew normal kernels for each cluster.
- `G`: an `ncol(res$data$Y)` $\times$ `ncol(res$data$Y)` $\times$ `res$prior$K` array storing the the posterior mean of the scaling matrix of the multivariate skew normal kernels for each cluster (in the parameterization used in the sampler).
- `E`: an `ncol(res$data$Y)` $\times$ `ncol(res$data$Y)` $\times$ `res$prior$K` array storing the the posterior mean of the covariance matrix of the multivariate normal distributions for each cluster form which the sample specific location parameters are drawn.
- `meanvec`: an `length(unique(res$data$C))` $\times$ `ncol(res$data$Y)` $\times$ `res$prior$K` array storing the the posterior mean of the multivariate skew normal kernels mean parameter for each sample and cluster.
- `meanvec0`: a `ncol(res$data$Y)` $\times$ `res$prior$K` matrix storing the posterior mean of the group mean parameter.
- `t`: Vector of length `nrow(x$data$Y)`. Each element is the mode of the posterior distribution of cluster labels.
- `eta`: scalar, the mean of the posterior distribution of the estimated Dirichlet Process Mixture concentration parameter.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )
```

```
# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)

# Generate calibrated data:
cal <- calibrateNoDist(res_relab)

# Compare raw and calibrated data: (see plot in vignette)
# par(mfrow=c(1, 2))
# plot(Y, col = C, xlim = range(Y[,1]), ylim = range(Y[,2]) )

# Get posterior estimates for the model parameters:
res_summary <- summarizeChain(res_relab)
# Check for instance, the cluster assignment labels:
table(res_summary$t)
```

```
# Indeed the same as
colSums(njk)

# Or examine the skewness parameter for the non-trivial clusters:
res_summary$alpha[ , unique(res_summary$t)]
# And compare those to
cbind(alpha1, alpha2)

# (see vignette for a more detailed example)
```

`tidyChain` *This function creates tidy versions of the stored chain. This object can then be used as input for the other diagnostic functions in this package.*

Description

This function creates tidy versions of the stored chain. This object can then be used as input for the other diagnostic functions in this package.

Usage

```
tidyChain(
  res,
  params = c("t", "w", "xi", "xi0", "psi", "G", "E", "eta", "Sigma", "alpha")
)
```

Arguments

`res`	An object of class COMIX.
`params`	A character vector naming the parameters to tidy.

Value

A `tidyChainCOMIX` object: a named list of class whose length is the length of `params`. Each element of the list contains a tibble with a tidy version of the samples from the MCMC chain.

Examples

```
library(COMIX)
# Number of observations for each sample (row) and cluster (column):
njk <-
  matrix(
    c(
      150, 300,
      250, 200
    ),
    nrow = 2,
    byrow = TRUE
  )
```

```
# Dimension of data:
p <- 3

# Scale and skew parameters for first cluster:
Sigma1 <- matrix(0.5, nrow = p, ncol = p) + diag(0.5, nrow = p)
alpha1 <- rep(0, p)
alpha1[1] <- -5
# location parameter for first cluster in first sample:
xi11 <- rep(0, p)
# location parameter for first cluster in second sample (aligned with first):
xi21 <- rep(0, p)

# Scale and skew parameters for second cluster:
Sigma2 <- matrix(-1/3, nrow = p, ncol = p) + diag(1 + 1/3, nrow = p)
alpha2 <- rep(0, p)
alpha2[2] <- 5
# location parameter for second cluster in first sample:
xi12 <- rep(3, p)
# location parameter for second cluster in second sample (misaligned with first):
xi22 <- rep(4, p)

# Sample data:
set.seed(1)
Y <-
  rbind(
    sn::rmsn(njk[1, 1], xi = xi11, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[1, 2], xi = xi12, Omega = Sigma2, alpha = alpha2),
    sn::rmsn(njk[2, 1], xi = xi21, Omega = Sigma1, alpha = alpha1),
    sn::rmsn(njk[2, 2], xi = xi22, Omega = Sigma2, alpha = alpha2)
  )

C <- c(rep(1, rowSums(njk)[1]), rep(2, rowSums(njk)[2]))

prior <- list(zeta = 1, K = 10)
pmc <- list(naprt = 5, nburn = 200, nsave = 200) # Reasonable usage
pmc <- list(naprt = 5, nburn = 2, nsave = 5) # Minimal usage for documentation
# Fit the model:
res <- comix(Y, C, pmc = pmc, prior = prior)

# Relabel to resolve potential label switching issues:
res_relab <- relabelChain(res)
tidy_chain <- tidyChain(res_relab)
# (see vignette for a more detailed example)
```

`transform_params` *Convert between parameterizations of the multivariate skew normal distribution.*

Description

Convert between parameterizations of the multivariate skew normal distribution.

Usage

```
transform_params(Sigma, alpha)
```

Arguments

Sigma	A scale matrix.
alpha	A vector for the skew parameter.

Value

A list:

- delta: a reparameterized skewness vector, a transformed version of alpha.
- omega: a diagonal matrix of the same dimensions as Sigma, the diagonal elements are the square roots of the diagonal elements of Sigma.
- psi: another reparameterized skewness vector, utilized in the sampler.
- G: a reparameterized version of Sigma, utilized in the sampler.

Examples

```
library(COMIX)
# Scale and skew parameters:
Sigma <- matrix(0.5, nrow = 4, ncol = 4) + diag(0.5, nrow = 4)
alpha <- c(0, 0, 0, 5)
transformed_parameters <- transform_params(Sigma, alpha)
```

Index