Package 'telescope'

June 15, 2024

Version 0.1-0

Encoding UTF-8

Title Bayesian Mixtures with an Unknown Number of Components

Description Fits Bayesian finite mixtures with an unknown number of components using the telescoping sampler and different component distributions. For more details see Frühwirth-Schnatter et al. (2021) <doi:10.1214/21-BA1294>.

Depends R (>= 4.0.0)

Imports abind, bayesm, DirichletReg, extraDistr, graphics, grDevices, MCMCpack, methods, mvtnorm, stats

VignetteBuilder knitr, rmarkdown

Suggests klaR, knitr, mclust, poLCA, rmarkdown

License GPL-2

RoxygenNote 7.3.1

NeedsCompilation no

Author Gertraud Malsiner-Walli [aut, cre] (<https://orcid.org/0000-0002-1213-4749>), Bettina Grün [aut] (<https://orcid.org/0000-0001-7265-4773>), Sylvia Frühwirth-Schnatter [aut] (<https://orcid.org/0000-0003-0516-5552>)

Maintainer Gertraud Malsiner-Walli <Gertraud.Malsiner-Walli@wu.ac.at>

Repository CRAN

Date/Publication 2024-06-15 09:00:02 UTC

Contents

`identifyMixture` *Solve label switching and identify mixture.*

Description

Clustering of the draws in the point process representation (PPR) using k-means clustering.

Usage

```
identifyMixture(Func, Mu, Eta, S, centers)
```

Arguments

`Func`	A numeric array of dimension $M \times d \times K$; data for clustering in the PPR.
`Mu`	A numeric array of dimension $M \times r \times K$; draws of cluster means.
`Eta`	A numeric array of dimension $M \times K$; draws of cluster sizes.
`S`	A numeric matrix of dimension $M \times N$; draws of cluster assignments.
`centers`	An integer or a numeric matrix of dimension $K \times d$; used to initialize `stats::kmeans()`.

Details

The following steps are implemented:

- A functional of the draws of the component-specific parameters (`Func`) is passed to the function. The functionals of each component and iteration are stacked on top of each other in order to obtain a matrix where each row corresponds to the functional of one component.
- The functionals are clustered into K_+ clusters using k-means clustering. For each functional a group label is obtained.
- The obtained labels of the functionals are used to construct a classification for each MCMC iteration. Those classifications which are a permutation of $(1, \ldots, K_+)$ are used to reorder the Mu and Eta draws and the assignment matrix S. This results in an identified mixture model.
- Note that only iterations resulting in permutations are used for parameter estimation and deriving the final partition. Those MCMC iterations where the obtained classifications of the functionals are not a permutation of $(1, \ldots, K_+)$ are discarded as no unique assignment of functionals to components can be made. If the non-permutation rate, i.e. the proportion of MCMC iterations where the obtained classifications of the functionals are not a permutation, is high, this is an indication of a poor clustering solution, as the functionals are not clearly separated.

Value

A named list containing:

- "S": reordered assignments.
- "Mu": reordered Mu matrix.
- "Eta": reordered weights.
- "non_perm_rate": proportion of draws where the clustering did not result in a permutation and hence no relabeling could be performed; this is the proportion of draws discarded.

---

plotBubble *Plot multivariate categorical data.*

Description

Plots of the level combinations of pairs of variables are created where the size of the circle indicating a level combination is proportional to the frequency of the level combination.

Usage

```
plotBubble(x, bg = "grey")
```

Arguments

x	A matrix or data.frame; the data consisting of categorical variables.
bg	If specified, the symbols are filled with colour(s), the vector bg is recycled to the number of observations. The default is to fill the symbols with grey color.

Value

NULL

Examples

```
with(chickwts, plotBubble(data.frame(cut(weight, 100 * 1:5), feed)))
```

plotScatter *Pairwise scatter plots of the data.*

Description

Scatter plots of the observations are plotted by selecting pairs of dimensions, potentially colored by a known classification.

Usage

```
plotScatter(x, z, label = "", trim = 0)
```

Arguments

x	A matrix or data.frame; the data consisting of metric variables.
z	A vector; indicating the color to use for the observations.
label	A character string; the text to include in the axes labels.
trim	A scalar numeric in $[0, 0.5)$; trimming to use for quantiles to determine axes.

Value

NULL

Examples

```
plotScatter(iris[, 1:4], iris$Species, label = "dim")
```

priorOnAlpha_spec *Specify prior on α.*

Description

Obtain a function to evaluate the log prior specified for α.

Usage

```
priorOnAlpha_spec(H = c("alpha_const", "gam_05_05", "gam_1_2", "F_6_3"))
```

Arguments

H	A character indicating which specification should be used.

Details

The following prior specifications are supported:

- `"alpha_const"`: α is fixed at 1.
- `"gam_05_05"`: $\alpha \sim$ gamma(0.5, 0.5), i.e., shape = 0.5, rate = 0.5.
- `"gam_1_2"`: $\alpha \sim$ gamma(1, 2), i.e., shape = 1, rate = 2.
- `"F_6_3"`: $\alpha \sim$ F(6, 3), i.e., an F-distribution with degrees of freedom equal to 6 and 3.

Value

A named list containing:

- `"log_pAlpha"`: a function of the log prior of α.
- `"param"`: a list with the parameters.

---

priorOnE0_spec *Specify prior on e0.*

Description

Obtain a function to evaluate the log prior specified for e_0.

Usage

```
priorOnE0_spec(E = c("G_1_20", "e0const"), e0)
```

Arguments

`E`	A character indicating which specification should be used.
`e0`	A numeric scalar giving the fixed value of e_0.

Details

The following prior specifications are supported:

- `"G_1_20"`: $e_0 \sim$ gamma(1, 20), i.e., shape = 1, rate = 20.
- `"e0const"`: e_0 is fixed at `e0`.

Value

A named list containing:

- `"log_p_e0"`: a function of the log prior of e_0.
- `"param"`: a list with the parameters.

`priorOnK_spec` *Specify prior on K.*

Description

Obtain a function to evaluate the log prior specified for K.

Usage

```
priorOnK_spec(
  P = c("fixedK", "Unif", "BNB_111", "BNB_121", "BNB_143", "BNB_443", "BNB_943",
   "Pois_1", "Pois_4", "Pois_9", "Geom_05", "Geom_02", "Geom_01", "NB_11", "NB_41",
    "NB_91"),
  K
)
```

Arguments

`P`	A character indicating which specification should be used. See Details for suitable values.
`K`	A numeric or integer scalar specifying the fixed (if `P` equals `"fixedK"`) or maximum value (if `P` equals `"Unif"`) of K.

Details

The following prior specifications are supported:

- `"fixedK"`: K has the fixed value K (second argument).
- `"Unif"`: $K \sim$ Unif$[1, K]$, where the upper limit is given by K (second argument).
- `"BNB_111"`: $K - 1 \sim$ BNB(1,1,1), i.e., $K - 1$ follows a beta-negative binomial distribution with parameters $(1, 1, 1)$.
- `"BNB_121"`: $K - 1 \sim$ BNB(1,2,1), i.e., $K - 1$ follows a beta-negative binomial distribution with parameters $(1, 2, 1)$.
- `"BNB_143"`: $K - 1 \sim$ BNB(1,2,1), i.e., $K - 1$ follows a beta-negative binomial distribution with parameters $(1, 4, 3)$.
- `"BNB_443"`: $K - 1 \sim$ BNB(4,4,3), i.e., $K - 1$ follows a beta-negative binomial distribution with parameters $(4, 4, 3)$.
- `"BNB_943"`: $K - 1 \sim$ BNB(9,4,3), i.e., $K - 1$ follows a beta-negative binomial distribution with parameters $(9, 4, 3)$.
- `"Pois_1"`: $K - 1 \sim$ pois(1), i.e., $K - 1$ follows a Poisson distribution with rate 1.
- `"Pois_4"`: $K - 1 \sim$ pois(4), i.e., $K - 1$ follows a Poisson distribution with rate 4.
- `"Pois_9"`: $K - 1 \sim$ pois(9), i.e., $K - 1$ follows a Poisson distribution with rate 9.
- `"Geom_05"`: $K - 1 \sim$ geom(0.5), i.e., $K - 1$ follows a geometric distribution with success probability $p = 0.5$ and density $f(x) = p(1 - p)^x$.

- `"Geom_02"`: $K - 1 \sim$ geom(0.2), i.e., $K - 1$ follows a geometric distribution with success probability $p = 0.2$ and density $f(x) = p(1-p)^x$.
- `"Geom_01"`: $K - 1 \sim$ geom(0.1), i.e., $K - 1$ follows a geometric distribution with success probability $p = 0.1$ and density $f(x) = p(1-p)^x$.
- `"NB_11"`: $K - 1 \sim$ nbinom(1,0.5), i.e., $K - 1$ follows a negative-binomial distribution with $size = 1$ and $p = 0.5$.
- `"NB_41"`: $K - 1 \sim$ nbinom(4,0.5), i.e., $K - 1$ follows a negative-binomial distribution with $size = 4$ and $p = 0.5$.
- `"NB_91"`: $K - 1 \sim$ nbinom(9,0.5), i.e., $K - 1$ follows a negative-binomial distribution with $size = 9$ and $p = 0.5$.

Value

A named list containing:

- `"log_pK"`: a function of the log prior of K.
- `"param"`: a list with the parameters.

`sampleAlpha` *Sample alpha conditional on partition and K using an Metropolis-Hastings step with log-normal proposal.*

Description

Sample α conditional on the current partition and value of K using an Metropolis-Hastings step with log-normal proposal.

Usage

```
sampleAlpha(N, Nk, K, alpha, s0_proposal, log_pAlpha)
```

Arguments

`N`	A number; indicating the sample size.
`Nk`	An integer vector; indicating the group sizes in the partition.
`K`	A number; indicating the number of components.
`alpha`	A numeric value; indicating the value for α.
`s0_proposal`	A numeric value; indicating the standard deviation of the random walk.
`log_pAlpha`	A function; evaluating the log prior of α.

Value

A named list containing:

- `"alpha"`: a numeric, the new α value.
- `"acc"`: logical indicating acceptance.

`sampleE0` *Sample e0 conditional on partition and K using an Metropolis-Hastings step with log-normal proposal.*

Description

Sample e_0 conditional on the current partition and value of K using an Metropolis-Hastings step with log-normal proposal.

Usage

```
sampleE0(K, Kp, N, Nk, s0_proposal, e0, log_p_e0)
```

Arguments

`K`	A number; indicating the number of components.
`Kp`	A number; indicating the number of filled components K_+.
`N`	A number; indicating the sample size.
`Nk`	An integer vector; indicating the group sizes in the partition.
`s0_proposal`	A numeric value; indicating the standard deviation of the random walk proposal.
`e0`	A numeric value; indicating the current value of e_0.
`log_p_e0`	A function; evaluating the log prior of e_0.

Value

A named list containing:

- `"e0"`: a numeric, the new e_0 value.
- `"acc"`: logical indicating acceptance.

`sampleK_alpha` *Sample K conditional on α where $e0 = \alpha/K$.*

Description

This sampling step only relies on the current partition and is independent of the current component-specific parameters, see Frühwirth-Schnatter et al (2021).

Usage

```
sampleK_alpha(Kp_j, Kmax, Nk_j, alpha, log_pK)
```

Arguments

Kp_j	A number; indicating the current value of K_+.
Kmax	A number; indicating the maximum value of K for which the conditional posterior is evaluated.
Nk_j	A numeric vector; indicating the group sizes in the partition, i.e., the current number of observations in the filled components.
alpha	A number; indicating the value of the parameter α.
log_pK	A function; evaluating the log prior of K.

Value

A number indicating the new value of K.

---

sampleK_e0 *Sample K conditional on e0 (fixed or random, but not depending on K).*

Description

This sampling step only relies on the current partition and is independent of the current component-specific parameters, see Frühwirth-Schnatter et al (2021).

Usage

```
sampleK_e0(Kp_j, Kmax, log_pK, log_p_e0, e0, N)
```

Arguments

Kp_j	A number; indicating the current value of K_+.
Kmax	A number; indicating the maximum value of K, for which the conditional posterior is evaluated.
log_pK	A function; evaluating the prior of K.
log_p_e0	A function; evaluating the log prior of e_0.
e0	A number; indicating the value of e_0.
N	A number; indicating the number of observations.

Value

A number indicating the new value of K.

`sampleLCA` *Telescoping sampling of the LCA model where a prior on the number of components K is specified.*

Description

- The MCMC scheme is implemented as suggested in Frühwirth-Schnatter et al (2021).
- The priors on the model parameters are specified as in Frühwirth-Schnatter et al (2021), see the vignette for details and notation.

Usage

```
sampleLCA(
  y,
  S,
  pi,
  eta,
  a0,
  M,
  burnin,
  thin,
  Kmax,
  G = c("MixDynamic", "MixStatic"),
  priorOnK,
  priorOnWeights,
  verbose = FALSE
)
```

Arguments

`y`	A numeric matrix; containing the data.
`S`	A numeric matrix; containing the initial cluster assignments.
`pi`	A numeric vector; containing the initial cluster-specific success probabilities.
`eta`	A numeric vector; containing the initial cluster sizes.
`a0`	A numeric vector; containing the parameters of the prior on the cluster-specific success probabilities.
`M`	A numeric scalar; specifying the number of recorded iterations.
`burnin`	A numeric scalar; specifying the number of burn-in iterations.
`thin`	A numeric scalar; specifying the thinning used for the iterations.
`Kmax`	A numeric scalar; the maximum number of components.
`G`	A character string; either `"MixDynamic"` or `"MixStatic"`.
`priorOnK`	A named list; providing the prior on the number of components K, see `priorOnK_spec()`.
`priorOnWeights`	A named list; providing the prior on the mixture weights.
`verbose`	A logical; indicating if some intermediate clustering results should be printed.

Value

A named list containing:

- `"Pi"`: sampled component-specific success probabilities.
- `"Eta"`: sampled weights.
- `"S"`: sampled assignments.
- `"Nk"`: number of observations assigned to the different components, for each iteration.
- `"K"`: sampled number of components.
- `"Kplus"`: number of filled, i.e., non-empty components, for each iteration.
- `"e0"`: sampled Dirichlet parameter of the prior on the weights (if e_0 is random).
- `"alpha"`: sampled Dirichlet parameter of the prior on the weights (if α is random).
- `"acc"`: logical vector indicating acceptance in the Metropolis-Hastings step when sampling either e_0 or α.

Examples

```
if (requireNamespace("poLCA", quietly = TRUE)) {
    data("carcinoma", package = "poLCA")
    y <- carcinoma
    N <- nrow(y)
    r <- ncol(y)

    Mmax <- 200
    thin <- 1
    burnin <- 100
    M <- Mmax/thin
    Kmax <- 50
    Kinit <- 10

    G <- "MixDynamic"
    priorOnAlpha <- priorOnAlpha_spec("gam_1_2")
    priorOnK <- priorOnK_spec("Pois_1")

    cat <- apply(y, 2, max)
    a0 <- rep(1, sum(cat))

    cl_y <- kmeans(y, centers = Kinit, iter.max = 20)
    S_0 <- cl_y$cluster
    eta_0 <- cl_y$size/N

    pi_0 <- do.call("cbind", lapply(1:r, function(j) {
        prop.table(table(S_0, y[, j]), 1)
    }))

    result <- sampleLCA(
        y, S_0, pi_0, eta_0, a0,
        M, burnin, thin, Kmax,
        G, priorOnK, priorOnAlpha)
```

```
    K <- result$K
    Kplus <- result$Kplus

    plot(seq_along(K), K, type = "l", ylim = c(0, max(K)),
         xlab = "iteration", main = "",
         ylab = expression("K" ~ "/" ~ K["+"]), col = 1)
    lines(seq_along(Kplus), Kplus, col = 2)
    legend("topright", legend = c("K", expression(K["+"])),
           col = 1:2, lty = 1, box.lwd = 0)
}
```

`sampleMultNormMixture` *Telescoping sampling of a Bayesian finite multivariate Gaussian mixture where a prior on the number of components is specified.*

Description

- The MCMC scheme is implemented as suggested in Frühwirth-Schnatter et al (2021).
- The priors on the model parameters are specified as in Frühwirth-Schnatter et al (2021), see the vignette for details and notation.
- The parameterizations of the Wishart and inverse Wishart distribution are used as in Frühwirth-Schnatter et al (2021), see also the vignette.

Usage

```
sampleMultNormMixture(
  y,
  S,
  mu,
  Sigma,
  eta,
  c0,
  g0,
  G0,
  C0,
  b0,
  B0,
  M,
  burnin,
  thin,
  Kmax,
  G = c("MixDynamic", "MixStatic"),
  priorOnK,
  priorOnWeights,
  verbose = FALSE
)
```

Arguments

y	A numeric matrix; containing the data.
S	A numeric matrix; containing the initial cluster assignments.
mu	A numeric matrix; containing the initial cluster-specific mean values.
Sigma	A numeric matrix; containing the initial cluster-specific variance covariance values.
eta	A numeric vector; containing the initial cluster sizes.
c0	A numeric vector; hyperparameter of the prior on Σ_k.
g0	A numeric vector; hyperparameter of the prior on C_0.
G0	A numeric vector; hyperparameter of the prior on C_0.
C0	A numeric vector; initial value of the hyperparameter C_0.
b0	A numeric vector; hyperparameter of the prior on μ_k.
B0	A numeric vector; hyperparameter of the prior on μ_k.
M	A numeric scalar; specifying the number of recorded iterations.
burnin	A numeric scalar; specifying the number of burn-in iterations.
thin	A numeric scalar; specifying the thinning used for the iterations.
Kmax	A numeric scalar; the maximum number of components.
G	A character string; either "MixDynamic" or "MixStatic".
priorOnK	A named list; providing the prior on the number of components K, see priorOnK_spec().
priorOnWeights	A named list; providing the prior on the mixture weights.
verbose	A logical; indicating if some intermediate clustering results should be printed.

Value

A named list containing:

- "Mu": sampled component means.
- "Eta": sampled weights.
- "S": sampled assignments.
- "Nk": number of observations assigned to the different components, for each iteration.
- "K": sampled number of components.
- "Kplus": number of filled, i.e., non-empty components, for each iteration.
- "e0": sampled Dirichlet parameter of the prior on the weights (if e_0 is random).
- "alpha": sampled Dirichlet parameter of the prior on the weights (if α is random).
- "acc": logical vector indicating acceptance in the Metropolis-Hastings step when sampling either e_0 or α.

Examples

```
y <- iris[, 1:4]
z <- iris$Species
r <- ncol(y)

Mmax <- 50
thin <- 1
burnin <- 0
M <- Mmax/thin
Kmax <- 40
Kinit <- 10

G <- "MixStatic"
priorOnE0 <- priorOnE0_spec("G_1_20", 1)
priorOnK <- priorOnK_spec("BNB_143")

R <- apply(y, 2, function(x) diff(range(x)))
b0 <- apply(y, 2, median)
B_0 <- rep(1, r)
B0 <- diag((R^2) * B_0)
c0 <- 2.5 + (r-1)/2
g0 <- 0.5 + (r-1)/2
G0 <- 100 * g0/c0 * diag((1/R^2), nrow = r)
C0 <- g0 * chol2inv(chol(G0))

cl_y <- kmeans(y, centers = Kinit, nstart = 100)
S_0 <- cl_y$cluster
mu_0 <- t(cl_y$centers)

eta_0 <- rep(1/Kinit, Kinit)
Sigma_0 <- array(0, dim = c(r, r, Kinit))
Sigma_0[, , 1:Kinit] <- 0.5 * C0

result <- sampleMultNormMixture(
  y, S_0, mu_0, Sigma_0, eta_0,
  c0, g0, G0, C0, b0, B0,
  M, burnin, thin, Kmax, G, priorOnK, priorOnE0)

K <- result$K
Kplus <- result$Kplus

plot(seq_along(K), K, type = "l", ylim = c(0, max(K)),
     xlab = "iteration", main = "",
     ylab = expression("K" ~ "/" ~ K["+"]), col = 1)
lines(seq_along(Kplus), Kplus, col = 2)
legend("topright", legend = c("K", expression(K["+"])),
       col = 1:2, lty = 1, box.lwd = 0)
```

`samplePoisMixture` *Telescoping sampling of a Bayesian finite Poisson mixture with a prior on the number of components K.*

Description

- The MCMC scheme is implemented as suggested in Frühwirth-Schnatter et al (2021).
- The priors on the model parameters are specified as in Frühwirth-Schnatter et al (2021) and Früwirth-Schnatter and Malsiner-Walli (2019), see the vignette for details and notation.

Usage

```
samplePoisMixture(
  y,
  S,
  mu,
  eta,
  a0,
  b0,
  h0,
  H0,
  M,
  burnin,
  thin,
  Kmax,
  G = c("MixDynamic", "MixStatic"),
  priorOnK,
  priorOnWeights,
  verbose = FALSE
)
```

Arguments

`y`	A numeric matrix; containing the data.
`S`	A numeric matrix; containing the initial cluster assignments.
`mu`	A numeric matrix; containing the initial cluster-specific rate values.
`eta`	A numeric vector; containing the initial cluster sizes.
`a0`	A numeric vector; hyperparameter of the prior on the rate μ.
`b0`	A numeric vector; hyperparameter of the prior on the rate μ.
`h0`	A numeric vector; hyperparameter of the prior on the rate μ.
`H0`	A numeric vector; hyperparameter of the prior on the rate μ.
`M`	A numeric scalar; specifying the number of recorded iterations.
`burnin`	A numeric scalar; specifying the number of burn-in iterations.
`thin`	A numeric scalar; specifying the thinning used for the iterations.
`Kmax`	A numeric scalar; the maximum number of components.

G	A character string; either "MixDynamic" or "MixStatic".
priorOnK	A named list; providing the prior on the number of components K, see priorOnK_spec().
priorOnWeights	A named list; providing the prior on the mixture weights.
verbose	A logical; indicating if some intermediate clustering results should be printed.

Value

A named list containing:

- "Mu": sampled rate μ.
- "Eta": sampled weights.
- "S": sampled assignments.
- "Nk": number of observations assigned to the different components, for each iteration.
- "K": sampled number of components.
- "Kplus": number of filled, i.e., non-empty components, for each iteration.
- "e0": sampled Dirichlet parameter of the prior on the weights (if e_0 is random).
- "alpha": sampled Dirichlet parameter of the prior on the weights (if α is random).
- "acc": logical vector indicating acceptance in the Metropolis-Hastings step when sampling either e0 or α.

Examples

```
N <- 200
z <- sample(1:2, N, prob = c(0.5, 0.5), replace = TRUE)
y <- rpois(N, c(1, 6)[z])

Mmax <- 200
thin <- 1
burnin <- 100
M <- Mmax/thin

Kmax <- 50
Kinit <- 10

G <- "MixDynamic"
priorOnAlpha <- priorOnAlpha_spec("gam_1_2")
priorOnK <- priorOnK_spec("BNB_143")

a0 <- 0.1
h0 <- 0.5
b0 <- a0/mean(y)
H0 <- h0/b0

cl_y <- kmeans(y, centers = Kinit, nstart = 100)
S_0 <- cl_y$cluster
mu_0 <- t(cl_y$centers)
eta_0 <- rep(1/Kinit, Kinit)
```

```
result <- samplePoisMixture(
  y, S_0, mu_0, eta_0,
  a0, b0, h0, H0,
  M, burnin, thin, Kmax,
  G, priorOnK, priorOnAlpha)

K <- result$K
Kplus <- result$Kplus

plot(seq_along(K), K, type = "l", ylim = c(0, max(K)),
     xlab = "iteration", main = "",
     ylab = expression("K" ~ "/" ~ K["+"]), col = 1)
lines(seq_along(Kplus), Kplus, col = 2)
legend("topright", legend = c("K", expression(K["+"])),
       col = 1:2, lty = 1, box.lwd = 0)
```

`sampleUniNormMixture` *Telescoping sampling of a Bayesian finite univariate Gaussian mixture where a prior on the number of components K is specified.*

Description

- The MCMC scheme is implemented as suggested in Frühwirth-Schnatter et al (2021).
- The priors on the model parameters are specified as in Frühwirth-Schnatter et al (2021), see the vignette for details and notation.
- The parametrizations of the gamma and inverse gamma distribution are used as in Frühwirth-Schnatter et al (2021), see also the vignette.

Usage

```
sampleUniNormMixture(
  y,
  S,
  mu,
  sigma2,
  eta,
  c0,
  g0,
  G0,
  C0_0,
  b0,
  B0,
  M,
  burnin,
  thin,
  Kmax,
  G = c("MixDynamic", "MixStatic"),
```

```
  priorOnK,
  priorOnWeights,
  verbose = FALSE
)
```

Arguments

y	A numeric matrix; containing the data.
S	A numeric matrix; containing the initial cluster assignments.
mu	A numeric matrix; containing the initial cluster-specific mean values.
sigma2	A numeric matrix; containing the initial cluster-specific variance values.
eta	A numeric vector; containing the initial cluster sizes.
c0	A numeric vector; hyperparameter of the prior on σ_k^2.
g0	A numeric vector; hyperparameter of the prior on σ_k^2
G0	A numeric vector; hyperparameter of the prior on σ_k^2
C0_0	A numeric vector; initial value of hyperparameter C_0.
b0	A numeric vector; hyperparameter of the prior on μ_k.
B0	A numeric vector; hyperparameter of the prior on μ_k.
M	A numeric scalar; specifying the number of recorded iterations.
burnin	A numeric scalar; specifying the number of burn-in iterations.
thin	A numeric scalar; specifying the thinning used for the iterations.
Kmax	A numeric scalar; the maximum number of components.
G	A character string; either "MixDynamic" or "MixStatic".
priorOnK	A named list; providing the prior on the number of components K, see priorOnK_spec().
priorOnWeights	A named list; providing the prior on the mixture weights.
verbose	A logical; indicating if some intermediate clustering results should be printed.

Value

A named list containing:

- "Mu": sampled component means.
- "Eta": sampled weights.
- "S": sampled assignments.
- "Nk": number of observations assigned to the different components, for each iteration.
- "K": sampled number of components.
- "Kplus": number of filled, i.e., non-empty components, for each iteration.
- "e0": sampled Dirichlet parameter of the prior on the weights (if e_0 is random).
- "alpha": sampled Dirichlet parameter of the prior on the weights (if α is random).
- "acc": logical vector indicating acceptance in the Metropolis-Hastings step when sampling either e_0 or α.

Examples

```
if (requireNamespace("mclust", quietly = TRUE)) {
    data("acidity", package = "mclust")
    y <- acidity

    N <- length(y)
    r <- 1

    Mmax <- 200
    thin <- 1
    burnin <- 100
    M <- Mmax/thin
    Kmax <- 50
    Kinit <- 10

    G <- "MixStatic"
    priorOnE0 <- priorOnE0_spec("e0const", 0.01)
    priorOnK <- priorOnK_spec("Pois_1", 50)

    R <- diff(range(y))
    c0 <- 2 + (r-1)/2
    C0 <- diag(c(0.02*(R^2)), nrow = r)
    g0 <- 0.2 + (r-1) / 2
    G0 <- diag(10/(R^2), nrow = r)
    B0 <- diag((R^2), nrow = r)
    b0 <- as.matrix((max(y) + min(y))/2, ncol = 1)

    cl_y <- kmeans(y, centers = Kinit, nstart = 100)
    S_0 <- cl_y$cluster
    mu_0 <- t(cl_y$centers)
    eta_0 <- rep(1/Kinit, Kinit)
    sigma2_0 <- array(0, dim = c(1, 1, Kinit))
    sigma2_0[1, 1, ] <- 0.5 * C0

    result <- sampleUniNormMixture(
        y, S_0, mu_0, sigma2_0, eta_0,
        c0, g0, G0, C0, b0, B0,
        M, burnin, thin, Kmax,
        G, priorOnK, priorOnE0)

    K <- result$K
    Kplus <- result$Kplus

    plot(seq_along(K), K, type = "l", ylim = c(0, max(K)),
         xlab = "iteration", main = "",
         ylab = expression("K" ~ "/" ~ K["+"]), col = 1)
    lines(seq_along(Kplus), Kplus, col = 2)
    legend("topright", legend = c("K", expression(K["+"])),
           col = 1:2, lty = 1, box.lwd = 0)
}
```

Index