Package 'BayesChange'

April 28, 2025

Title Bayesian Methods for Change Points Analysis

Version 2.1.0

Description

Perform change points detection on univariate and multivariate time series according to the methods presented by Asael Fabian Martínez and Ramsés H. Mena (2014) <doi:10.1214/14-BA878> and Corradin, Danese and Ongaro (2022) <doi:10.1016/j.ijar.2021.12.019>. It also clusters different types of time dependent data with common change points, see ``Model-based clustering of time-dependent observations with common structural changes'' (Corradin,Danese,KhudaBukhsh and Ongaro, 2024) <doi:10.48550/arXiv.2410.09552> for details.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp, RcppArmadillo, RcppGSL

Imports Rcpp, salso, dplyr, tidyr, ggplot2, ggpubr

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL `https://github.com/lucadanese/BayesChange`

BugReports `https://github.com/lucadanese/BayesChange/issues`

NeedsCompilation yes

Author Luca Danese [aut, cre, cph] (<`https://orcid.org/0000-0001-8444-8563`>),
Riccardo Corradin [aut],
Andrea Ongaro [aut]

Maintainer Luca Danese <`l.danese1@campus.unimib.it`>

Repository CRAN

Date/Publication 2025-04-28 11:00:02 UTC

Contents

`ClustCpObj` *ClustCpObj class constructor*

Description

A constructor for the `ClustCpObj` class. The class `ClustCpObj` contains...

Usage

```
ClustCpObj(
  data = NULL,
  n_iterations = NULL,
  n_burnin = NULL,
  clust = NULL,
  orders = NULL,
  time = NULL,
  lkl = NULL,
  norm_vec = NULL,
  I0_MCMC = NULL,
  I0_MCMC_01 = NULL,
  kernel_ts = NULL,
  kernel_epi = NULL,
  univariate_ts = NULL
)
```

Arguments

data	a vector or a matrix containing the values of the time series;
n_iterations	number of iterations of the MCMC algorithm;
n_burnin	number of MCMC iterations to exclude in the posterior estimate;
clust	a matrix with the clustering of each iteration.
orders	a matrix where each row corresponds to the output order of the corresponding iteration;
time	computational time in seconds;
lkl	a vector with the likelihood of the final clustering.
norm_vec	a vector with the estimated normalization constant.
I0_MCMC	traceplot for I_0.
I0_MCMC_01	a $0/1$ vector, the n-th element is equal to 1 if the proposed I_0 was accepted, 0 otherwise.
kernel_ts	if TRUE data are time series.
kernel_epi	if TRUE data are survival functions.
univariate_ts	TRUE/FALSE if time series is univariate or not;

---

clust_cp *Clustering time dependent observations with common change points.*

Description

The clust_cp function cluster observations with common change points. Data can be time series or survival functions.

Usage

```
clust_cp(
  data,
  n_iterations,
  n_burnin = 0,
  params = list(),
  alpha_SM = 1,
  B = 1000,
  L = 1,
  q = 0.5,
  kernel,
  print_progress = TRUE,
  user_seed = 1234
)
```

Arguments

- `data` — a matrix or an array If a matrix the algorithm for univariate time series is used, where each row is a time series. If an array, the algorithm is run for multivariate time series. Each slice of the array is a matrix where the rows are the dimensions of the time series.
- `n_iterations` — number of MCMC iterations.
- `n_burnin` — number of iterations that must be excluded when computing the posterior estimate.
- `params` — a list of parameters:

  If the time series is univariate the following must be specified:

  - `a`,`b`,`c` parameters of the integrated likelihood.
  - `phi` correlation parameter in the likelihood.

  If the time series is multivariate the following must be specified:

  - `k_0`,`nu_0`,`S_0`,`m_0` parameters of the integrated likelihood.
  - `phi` correlation parameter in the likelihood.

  If data are survival functions:

  - `M` number of Monte Carlo iterations when computing the likelihood of the survival function.
  - `xi` recovery rate fixed constant for each population at each time.
  - `a0`,`b0` parameters for the computation of the integrated likelihood of the survival functions.
  - `I0_var` variance for the Metropolis-Hastings estimation of the proportion of infected at time 0.
  - `p` prior average number of change points for each order.
- `alpha_SM` — α for the split-merge main algorithm.
- `B` — number of orders for the normalization constant.
- `L` — number of split-merge steps for the proposal step.
- `q` — probability of a split in the split-merge proposal and acceleration step.
- `kernel` — can be "ts" if data are time series or "epi" if data are survival functions.
- `print_progress` — If TRUE (default) print the progress bar.
- `user_seed` — seed for random distribution generation.

Value

A `ClustCpObj` class object containing

- `$data` vector or matrix with the data.
- `$n_iterations` number of iterations.
- `$n_burnin` number of burn-in iterations.
- `$clust` a matrix where each row corresponds to the output cluster of the corresponding iteration.

- `$orders` a multidimensional array where each slice is a matrix and represent an iteration. The row of each matrix correspond the order associated to the corresponding cluster.
- `$time` computational time.
- `$lkl` a matrix where each row is the likelihood of each observation computed at the corresponding iteration.
- `$norm_vec` a vector containing the normalization constant computed at the beginning of the algorithm.
- `I0_MCMC` traceplot for I_0.
- `I0_MCMC_01` a $0/1$ vector, the n-th element is equal to 1 if the proposed I_0 was accepted, 0 otherwise.
- `$kernel_ts` if TRUE data are time series.
- `$kernel_epi` if TRUE data are survival function.
- `$univariate_ts` TRUE if data is an univariate time series, FALSE if it is a multivariate time series.

References

Corradin, R., Danese, L., KhudaBukhsh, W. R., & Ongaro, A. (2024). *Model-based clustering of time-dependent observations with common structural changes*. arXiv preprint arXiv:2410.09552.

Examples

```
## Univariate time series

data_mat <- matrix(NA, nrow = 5, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_mat[4,] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_mat[5,] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp(data = data_mat, n_iterations = 5000, n_burnin = 1000,
                 L = 1, params = list(phi = 0.5), B = 1000, kernel = "ts")

print(out)

## Multivariate time series


data_array <- array(data = NA, dim = c(3,100,5))

data_array[1,,1] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[2,,1] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[3,,1] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))

data_array[1,,2] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[2,,2] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
```

```
data_array[3,,2] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))

data_array[1,,3] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_array[2,,3] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_array[3,,3] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))

data_array[1,,4] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_array[2,,4] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_array[3,,4] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))

data_array[1,,5] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))
data_array[2,,5] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))
data_array[3,,5] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp(data = data_array, n_iterations = 3000, n_burnin = 1000,
                params = list(phi = 0.5, k_0 = 0.25,
                              nu_0 = 5, S_0 = diag(0.1,3,3),
                              m_0 = rep(0,3)), B = 1000,  kernel = "ts")

print(out)

## Epidemiological data



data_mat <- matrix(NA, nrow = 5, ncol = 50)

betas <- list(c(rep(0.45, 25),rep(0.14,25)),
              c(rep(0.55, 25),rep(0.11,25)),
              c(rep(0.50, 25),rep(0.12,25)),
              c(rep(0.52, 10),rep(0.15,40)),
              c(rep(0.53, 10),rep(0.13,40)))

inf_times <- list()

for(i in 1:5){

  inf_times[[i]] <- sim_epi_data(10000, 10, 50, betas[[i]], 1/8)

  vec <- rep(0,50)
  names(vec) <- as.character(1:50)

  for(j in 1:50){
    if(as.character(j) %in% names(table(floor(inf_times[[i]])))){
    vec[j] = table(floor(inf_times[[i]]))[which(names(table(floor(inf_times[[i]]))) == j)]
    }
  }
  data_mat[i,] <- vec
}

out <- clust_cp(data = data_mat, n_iterations = 100, n_burnin = 10,
                params = list(M = 100, xi = 1/8), B = 1000, kernel = "epi")
```

```
print(out)
```

`clust_cp_epi` *Clustering Epidemiological survival functions with common changes in time*

Description

Clustering Epidemiological survival functions with common changes in time

Usage

```
clust_cp_epi(
  data,
  n_iterations,
  M,
  B,
  L,
  xi = 1/8,
  alpha_SM = 1,
  q = 0.1,
  a0 = 4,
  b0 = 10,
  I0_var = 0.01,
  avg_blk = 0.003,
  print_progress = TRUE,
  user_seed = 1234L
)
```

Arguments

`data`	a matrix where each entry is the number of infected for a population (row) at a specific discrete time (column).
`n_iterations`	Second value
`M`	number of Monte Carlo iterations when computing the likelihood of the survival function.
`B`	number of orders for the normalisation constant.
`L`	number of split-merge steps for the proposal step.
`xi`	recovery rate fixed constant for each population at each time.
`alpha_SM`	α parameter for the main split-merge algorithm.
`q`	probability of performing a split when updating the single order for the proposal procedure.

a0, b0	parameters for the computation of the integrated likelihood of the survival functions.
I0_var	variance for the Metropolis-Hastings estimation of the proportion of infected at time 0.
avg_blk	average number of change points for the random generated orders.
print_progress	If TRUE (default) print the progress bar.
user_seed	seed for random distribution generation.

Value

Function clust_cp_epi returns a list containing the following components:

- $clust a matrix where each row corresponds to the output cluster of the corresponding iteration.
- $orders a multidimensional matrix where each slice is a matrix with the orders associated to the output cluster of that iteration.
- time computational time in seconds.
- $llik a matrix containing the log-likelihood of each population at each iteration.
- $rho traceplot for the proportion of infected individuals at time 0.

Examples

```
data_mat <- matrix(NA, nrow = 5, ncol = 50)

betas <- list(c(rep(0.45, 25),rep(0.14,25)),
              c(rep(0.55, 25),rep(0.11,25)),
              c(rep(0.50, 25),rep(0.12,25)),
              c(rep(0.52, 10),rep(0.15,40)),
              c(rep(0.53, 10),rep(0.13,40)))

 inf_times <- list()

 for(i in 1:5){

   inf_times[[i]] <- sim_epi_data(10000, 10, 50, betas[[i]], 1/8)

   vec <- rep(0,50)
   names(vec) <- as.character(1:50)

   for(j in 1:50){
     if(as.character(j) %in% names(table(floor(inf_times[[i]])))){
     vec[j] = table(floor(inf_times[[i]]))[which(names(table(floor(inf_times[[i]]))) == j)]
     }
   }
   data_mat[i,] <- vec
 }

 out <- clust_cp_epi(data = data_mat, n_iterations = 3000, M = 250, B = 1000, L = 1)
```

clust_cp_multi *Clustering multivariate times series with common changes in time*

Description

Clustering multivariate times series with common changes in time

Usage

```
clust_cp_multi(
  data,
  n_iterations,
  B,
  L,
  phi,
  k_0,
  nu_0,
  S_0,
  m_0,
  q = 0.5,
  alpha_SM = 0.1,
  print_progress = TRUE,
  user_seed = 1234L
)
```

Arguments

`data`	a multidimensional matrix where each element is a matrix whose rows are the observations and columns the dimensions.
`n_iterations`	number of MCMC iterations.
`B`	number of orders for the normalisation constant.
`L`	number of split-merge steps for the proposal step.
`phi`, `k_0`, `nu_0`, `S_0`, `m_0`	parameters of the integrated likelihood.
`q`	probability of a split in the split-merge proposal and acceleration step.
`alpha_SM`	α for the main split-merge algorithm.
`print_progress`	If TRUE (default) print the progress bar.
`user_seed`	seed for random distribution generation.

Value

Function `clust_cp_multi` returns a list containing the following components:

- `$clust` a matrix where each row corresponds to the output cluster of the corresponding iteration.

- `$orders` a multidimensional array where each slice is a matrix and represent an iteration. The row of each matrix correspond the order associated to the corresponding cluster.
- `time` computational time in seconds.
- `$norm_vec` a vector containing the normalisation constant computed at the beginning of the algorithm.

Examples

```
data_array <- array(data = NA, dim = c(3,100,5))

data_array[1,,1] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[2,,1] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[3,,1] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))

data_array[1,,2] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[2,,2] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_array[3,,2] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))

data_array[1,,3] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_array[2,,3] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_array[3,,3] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))

data_array[1,,4] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_array[2,,4] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_array[3,,4] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))

data_array[1,,5] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))
data_array[2,,5] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))
data_array[3,,5] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp_multi(data = data_array, n_iterations = 3000, B = 1000, L = 1,
                      phi = 0.1, k_0 = 0.25, nu_0 = 5, S_0 = diag(0.1,3,3), m_0 = rep(0,3))
```

`clust_cp_uni` *Clustering univariate times series with common changes in time*

Description

Clustering univariate times series with common changes in time

Usage

```
clust_cp_uni(
  data,
  n_iterations,
  B,
  L,
  phi,
```

```
  a = 1,
  b = 1,
  c = 1,
  q = 0.5,
  alpha_SM = 0.1,
  print_progress = TRUE,
  user_seed = 1234L
)
```

Arguments

`data`	a matrix where each row is an observation and each column corresponds to a discrete time.
`n_iterations`	number of MCMC iterations.
`B`	number of orders for the normalisation constant.
`L`	number of split-merge steps for the proposal step.
`phi`, `a`, `b`, `c`	parameters of the integrated likelihood.
`q`	probability of a split in the split-merge proposal and acceleration step.
`alpha_SM`	α for the main split-merge algorithm.
`print_progress`	If TRUE (default) print the progress bar.
`user_seed`	seed for random distribution generation.

Value

Function `clust_cp_uni` returns a list containing the following components:

- `$clust` a matrix where each row corresponds to the output cluster of the corresponding iteration.
- `$orders` a multidimensional array where each slice is a matrix and represent an iteration. The row of each matrix correspond the order associated to the corresponding cluster.
- `$time` computational time in seconds.
- `$norm_vec` a vector containing the normalisation constant computed at the beginning of the algorithm.

Examples

```
data_mat <- matrix(NA, nrow = 5, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_mat[4,] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_mat[5,] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp_uni(data = data_mat, n_iterations = 5000, B = 1000, L = 1, phi = 0.5)
```

DetectCpObj *DetectCpObj class constructor*

Description

A constructor for the DetectCpObj class. The class DetectCpObj contains...

Usage

```
DetectCpObj(
  data = NULL,
  n_iterations = NULL,
  n_burnin = NULL,
  orders = NULL,
  time = NULL,
  phi_MCMC = NULL,
  phi_MCMC_01 = NULL,
  sigma_MCMC = NULL,
  sigma_MCMC_01 = NULL,
  theta_MCMC = NULL,
  I0_MCMC = NULL,
  I0_MCMC_01 = NULL,
  kernel_ts = NULL,
  kernel_epi = NULL,
  univariate_ts = NULL
)
```

Arguments

data	a vector or a matrix containing the values of the time series;
n_iterations	number of iterations of the MCMC algorithm;
n_burnin	number of MCMC iterations to exclude in the posterior estimate;
orders	a matrix where each row corresponds to the output order of the corresponding iteration;
time	computational time in seconds;
phi_MCMC	traceplot for γ.
phi_MCMC_01	a $0/1$ vector, the n-th element is equal to 1 if the proposed ϕ was accepted, 0 otherwise.
sigma_MCMC	traceplot for σ.
sigma_MCMC_01	a $0/1$ vector, the n-th element is equal to 1 if the proposed σ was accepted, 0 otherwise.
theta_MCMC	traceplot for θ.
I0_MCMC	traceplot for I_0.

I0_MCMC_01	a $0/1$ vector, the n-th element is equal to 1 if the proposed I_0 was accepted, 0 otherwise.
kernel_ts	if TRUE data are time series.
kernel_epi	if TRUE data are survival functions.
univariate_ts	TRUE/FALSE if time series is univariate or not.

---

detect_cp *Detect change points on time series.*

Description

The detect_cp function detect change points on univariate and multivariate time series.

Usage

```
detect_cp(
  data,
  n_iterations,
  n_burnin = 0,
  q = 0.5,
  params = list(),
  kernel,
  print_progress = TRUE,
  user_seed = 1234
)
```

Arguments

data
: a vector or a matrix. If a vector the algorithm for univariate time series is used. If a matrix, where rows are the observations and columns are the times, then the algorithm for multivariate time series is used.

n_iterations
: number of MCMC iterations.

n_burnin
: number of iterations that must be excluded when computing the posterior estimate.

q
: probability of performing a split at each iteration.

params
: a list of parameters:

If data is an univariate time series the following must be specified:

- a, b, c parameters of the Normal-Gamma prior for μ and λ.
- prior_var_phi variance for the proposal in the $N(0, \sigma^2_\phi)$ posterior estimate of θ.
- par_theta_c parameter of the shifted Gamma prior of θ.
- par_theta_d parameter of the shifted Gamma prior of θ.

If the time series is multivariate the following must be specified:

- m_0, k_0, nu_0, S_0 parameters for the Normal-Inverse-Wishart prior for (μ, λ).
- prior_var_phi variance for the proposal in the $N(0, \sigma_\phi^2)$ posterior estimate of θ.
- par_theta_c parameter of the shifted Gamma prior of θ.
- par_theta_d parameter of the shifted Gamma prior of θ.

If data are survival functions:

- M number of Monte Carlo iterations when computing the likelihood of the survival function.
- xi recovery rate fixed constant for each population at each time.
- a0,b0 parameters for the computation of the integrated likelihood of the survival functions.
- I0_var variance for the Metropolis-Hastings estimation of the proportion of infected at time 0.
- p prior average number of change points for each order.

kernel can be "ts" if data are time series or "epi" if data are survival functions.

print_progress If TRUE (default) print the progress bar.

user_seed seed for random distribution generation.

Value

A DetectCpObj class object containing

- $data vector or matrix with the data.
- $n_iterations number of iterations.
- $n_burnin number of burn-in iterations.
- $orders matrix where each entries is the assignment of the realization to a block. Rows are the iterations and columns the times.
- $time computational time.
- $gammaMCMC traceplot for γ.
- $phi_MCMC_01 a $0/1$ vector, the n-th element is equal to 1 if the proposed γ was accepted, 0 otherwise.
- $sigma_MCMC traceplot for σ.
- $sigma_MCMC_01 a $0/1$ vector, the n-th element is equal to 1 if the proposed σ was accepted, 0 otherwise.
- $theta_MCMC traceplot for θ.
- I0_MCMC traceplot for I_0.
- I0_MCMC_01 a $0/1$ vector, the n-th element is equal to 1 if the proposed I_0 was accepted, 0 otherwise.
- kernel_ts if TRUE data are time series.
- kernel_epi if TRUE data are survival functions.
- $univariate_ts TRUE if data is an univariate time series, FALSE if it is a multivariate time series.

References

Martínez, A. F., & Mena, R. H. (2014). On a Nonparametric Change Point Detection Model in Markovian Regimes. *Bayesian Analysis*, 9(4), 823–858. doi:10.1214/14BA878

Corradin, R., Danese, L., & Ongaro, A. (2022). Bayesian nonparametric change point detection for multivariate time series with missing observations. *International Journal of Approximate Reasoning*, 143, 26–43. doi:10.1016/j.ijar.2021.12.019

Corradin, R., Danese, L., KhudaBukhsh, W. R., & Ongaro, A. (2024). *Model-based clustering of time-dependent observations with common structural changes*. arXiv preprint arXiv:2410.09552.

Examples

```
## Univariate time series

data_vec <- as.numeric(c(rnorm(50,0,0.1), rnorm(50,1,0.25)))

out <- detect_cp(data = data_vec, n_iterations = 2500, n_burnin = 500,
                 params = list(a = 1, b = 1, c = 0.1), kernel = "ts")

print(out)

## Multivariate time series

data_mat <- matrix(NA, nrow = 3, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))

out <- detect_cp(data = data_mat, n_iterations = 2500, n_burnin = 500,
                 params = list(m_0 = rep(0,3), k_0 = 0.25, nu_0 = 4,
                               S_0 = diag(1,3,3)), kernel = "ts")

print(out)


## Survival functions

data_mat <- matrix(NA, nrow = 1, ncol = 100)

betas <- c(rep(0.45, 25),rep(0.14,75))

inf_times <- sim_epi_data(10000, 10, 100, betas, 1/8)

inf_times_vec <- rep(0,100)
names(inf_times_vec) <- as.character(1:100)

for(j in 1:100){
 if(as.character(j) %in% names(table(floor(inf_times)))){
   inf_times_vec[j] = table(floor(inf_times))[which(names(table(floor(inf_times))) == j)]
 }
}
```

```
data_mat[1,] <- inf_times_vec

out <- detect_cp(data = data_mat, n_iterations = 500, n_burnin = 100,
                 params = list(M = 250, xi = 1/8, a0 = 40, b0 = 10), kernel = "epi")

print(out)
```

---

detect_cp_epi *Detect Change Points on a survival function*

Description

Detect Change Points on a survival function

Usage

```
detect_cp_epi(
  data,
  n_iterations,
  q,
  M,
  xi,
  a0,
  b0,
  I0_var = 0.01,
  print_progress = TRUE,
  user_seed = 1234L
)
```

Arguments

data	a matrix where each row is a component of the time series and the columns correspond to the times.
n_iterations	number of MCMC iterations.
q	probability of performing a split at each iteration.
M	number of Monte Carlo iterations when computing the likelihood of the survival function.
xi	recovery rate fixed constant for each population at each time.
a0, b0	parameters for the computation of the integrated likelihood of the survival functions.
I0_var	variance for the Metropolis-Hastings estimation of the proportion of infected at time 0.
print_progress	If TRUE (default) print the progress bar.
user_seed	seed for random distribution generation.

Value

Function `detect_cp_epi` returns a list containing the following components:

- `$orders` a matrix where each row corresponds to the output order of the corresponding iteration.
- `time` computational time in seconds.
- `$I0_MCMC` traceplot for I_0.
- `$I0_MCMC_01` a $0/1$ vector, the n-th element is equal to 1 if the proposed I_0 was accepted, 0 otherwise.

Examples

```
data_mat <- matrix(NA, nrow = 1, ncol = 100)

betas <- c(rep(0.45, 25),rep(0.14,75))

inf_times <- sim_epi_data(10000, 10, 100, betas, 1/8)

inf_times_vec <- rep(0,100)
names(inf_times_vec) <- as.character(1:100)

for(j in 1:100){
 if(as.character(j) %in% names(table(floor(inf_times)))){
 inf_times_vec[j] =
 table(floor(inf_times))[which(names(table(floor(inf_times))) == j)]}
}

data_mat[1,] <- inf_times_vec

out <- detect_cp_epi(data = data_mat, n_iterations = 250, q = 0.5,
                     xi = 1/8, a0 = 40, b0 = 10, M = 250)
```

`detect_cp_multi` *Detect Change Points on multivariate time series*

Description

Detect Change Points on multivariate time series

Usage

```
detect_cp_multi(
  data,
  n_iterations,
  q,
  k_0,
```

```
    nu_0,
    S_0,
    m_0,
    par_theta_c = 1,
    par_theta_d = 1,
    prior_var_phi = 0.1,
    print_progress = TRUE,
    user_seed = 1234L
)
```

Arguments

data	a matrix where each row is a component of the time series and the columns correpospond to the times.
n_iterations	number of MCMC iterations.
q	probability of performing a split at each iteration.
k_0, nu_0, S_0, m_0	parameters for the Normal-Inverse-Wishart prior for (μ, λ).
par_theta_c, par_theta_d	parameters for the shifted Gamma prior for θ.
prior_var_phi	parameters for the correlation coefficient in the likelihood.
print_progress	If TRUE (default) print the progress bar.
user_seed	seed for random distribution generation.

Value

Function detect_cp_multi returns a list containing the following components:

- $orders a matrix where each row corresponds to the output order of the corresponding iteration.
- time computational time in seconds.
- $phi_MCMC traceplot for γ.
- $phi_MCMC_01 a $0/1$ vector, the n-th element is equal to 1 if the proposed ϕ was accepted, 0 otherwise.
- $sigma_MCMC traceplot for σ.
- $sigma_MCMC_01 a $0/1$ vector, the n-th element is equal to 1 if the proposed σ was accepted, 0 otherwise.
- $theta_MCMC traceplot for θ.

Examples

```
data_mat <- matrix(NA, nrow = 3, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
```

```
out <- detect_cp_multi(data = data_mat,
                              n_iterations = 2500,
                         q = 0.25,k_0 = 0.25, nu_0 = 4, S_0 = diag(1,3,3), m_0 = rep(0,3),
                              par_theta_c = 2, par_theta_d = 0.2, prior_var_phi = 0.1)
```

`detect_cp_uni` *Detect Change Points on an univariate time series.*

Description

Detect Change Points on an univariate time series.

Usage

```
detect_cp_uni(
  data,
  n_iterations,
  q,
  a = 1,
  b = 1,
  c = 0.1,
  prior_var_phi = 0.1,
  par_theta_c = 1,
  par_theta_d = 1,
  print_progress = TRUE,
  user_seed = 1234L
)
```

Arguments

data	vector of observations.
n_iterations	number of MCMC iteration.
q	probability of performing a split at each iterations.
a, b, c	parameters of the Normal-Gamma prior for μ and λ.
prior_var_phi	parameters for the correlation coefficient in the likelihood.
par_theta_c, par_theta_d	parameters of the shifted Gamma prior for θ.
print_progress	If TRUE (default) print the progress bar.
user_seed	seed for random distribution generation.

Value

Function detect_cp_uni returns a list containing the following components:

- $orders a matrix where each row corresponds to the output order of the corresponding iteration.
- time computational time in seconds.
- $sigma_MCMC traceplot for σ.
- $sigma_MCMC_01 a $0/1$ vector, the n-th element is equal to 1 if the proposed σ was accepted, 0 otherwise.
- $theta_MCMC traceplot for θ.

Examples

```
data_vec <- as.numeric(c(rnorm(50,0,0.1), rnorm(50,1,0.25)))

out <- detect_cp_uni(data = data_vec,
                            n_iterations = 2500,
                            q = 0.25)
```

---

plot.ClustCpObj *Plot estimated partition*

Description

The plot method plots the estimates partition through the salso algorithm, for a ClustCpObj class object.

Usage

```
## S3 method for class 'ClustCpObj'
plot(
  x,
  y = NULL,
  loss = "VI",
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  ...
)
```

Arguments

x	an object of class ClustCpObj.
y	parameter of the generic method.
loss	The loss function used to estimate the final partition, it can be "VI", "binder", "omARI", "NVI", "ID", "NID".
maxNClusters	maximum number of clusters in salso procedure.
nRuns	number of runs in salso procedure.
maxZealousAttempts	maximum number of zealous attempts in salso procedure.
...	parameter of the generic method.

Value

The function returns a ggplot object representing the time series or the survival functions colored according to the final partition.

Examples

```
## Time series

data_mat <- matrix(NA, nrow = 5, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_mat[4,] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_mat[5,] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp(data = data_mat, n_iterations = 5000, n_burnin = 1000,
                params = list(L = 1, B = 1000, phi = 0.5), kernel = "ts")

plot(out)


## Survival functions

data_mat <- matrix(NA, nrow = 5, ncol = 50)

betas <- list(c(rep(0.45, 25),rep(0.14,25)),
              c(rep(0.55, 25),rep(0.11,25)),
              c(rep(0.50, 25),rep(0.12,25)),
              c(rep(0.52, 10),rep(0.15,40)),
              c(rep(0.53, 10),rep(0.13,40)))

inf_times <- list()

for(i in 1:5){
  inf_times[[i]] <- sim_epi_data(10000, 10, 50, betas[[i]], 1/8)
```

```
  vec <- rep(0,50)
  names(vec) <- as.character(1:50)
  for(j in 1:50){
    if(as.character(j) %in% names(table(floor(inf_times[[i]])))){
    vec[j] = table(floor(inf_times[[i]]))[which(names(table(floor(inf_times[[i]]))) == j)]
    }
  }
  data_mat[i,] <- vec
}

out <- clust_cp(data = data_mat, n_iterations = 100, n_burnin = 10,
                params = list(M = 100, L = 1, B = 100), kernel = "epi")

plot(out)
```

---

plot.DetectCpObj   *Plot estimated change points*

Description

The plot method plots the estimates change points estimated through the salso algorithm, for a DetectCpObj class object.

Usage

```
## S3 method for class 'DetectCpObj'
plot(
  x,
  y = NULL,
  plot_freq = FALSE,
  loss = "VI",
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  ...
)
```

Arguments

x	an object of class DetectCPObj.
y, ...	parameters of the generic method.
plot_freq	if TRUE also the histogram with the empirical frequency of each change point is plotted.
loss	The loss function used to estimate the final partition, it can be "VI", "binder", "omARI", "NVI", "ID", "NID".

`maxNClusters` maximum number of clusters in salso procedure.

`nRuns` number of runs in salso procedure.

`maxZealousAttempts` maximum number of zealous attempts in salso procedure.

Value

The function returns a ggplot object representing the detected change points. If `plot_freq = TRUE` is plotted also an histogram with the frequency of times that a change point has been detected in the MCMC chain.

Examples

```
data_mat <- matrix(NA, nrow = 3, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))

out <- detect_cp(data = data_mat, n_iterations = 2500, n_burnin = 500,
                 params = list(q = 0.25, k_0 = 0.25, nu_0 = 4, S_0 = diag(1,3,3),
                               m_0 = rep(0,3), par_theta_c = 2, par_theta_d = 0.2,
                               prior_var_phi = 0.1), kernel = "ts")
plot(out)
```

---

posterior_estimate.ClustCpObj

Estimate the change points of the data

Description

The `posterior_estimate` method estimates the change points of the data making use of the salso algorithm, for a `DetectCPObj` class object.

Usage

```
## S3 method for class 'ClustCpObj'
posterior_estimate(
  object,
  loss = "VI",
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  ...
)
```

Arguments

`object`	an object of class `ClustCpObj`.
`loss`	The loss function used to estimate the final partition, it can be "VI", "binder", "omARI", "NVI", "ID", "NID".
`maxNClusters`	maximum number of clusters in salso procedure.
`nRuns`	number of runs in salso procedure.
`maxZealousAttempts`	maximum number of zealous attempts in salso procedure.
`...`	parameter of the generic method.

Details

put details here

Value

The function returns a vector with the cluster assignment of each observation.

References

#' D. B. Dahl, D. J. Johnson, and P. Müller (2022), Search Algorithms and Loss Functions for Bayesian Clustering, *Journal of Computational and Graphical Statistics*, 31(4), 1189-1201, doi:10.1080/10618600.2022.2069779.

Examples

```
data_mat <- matrix(NA, nrow = 5, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_mat[4,] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_mat[5,] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp(data = data_mat, n_iterations = 5000, n_burnin = 1000,
                params = list(L = 1, B = 1000, phi = 0.5), kernel = "ts")

posterior_estimate(out)
```

posterior_estimate.DetectCpObj
Estimate the change points of the data

Description

The `posterior_estimate` method estimates the change points of the data making use of the salso algorithm, for a `DetectCPObj` class object.

Usage

```
## S3 method for class 'DetectCpObj'
posterior_estimate(
  object,
  loss = "VI",
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  ...
)
```

Arguments

`object`	an object of class `DetectCPObj`.
`loss`	The loss function used to estimate the final partition, it can be "VI", "binder", "omARI", "NVI", "ID", "NID".
`maxNClusters`	maximum number of clusters in salso procedure.
`nRuns`	number of runs in salso procedure.
`maxZealousAttempts`	maximum number of zealous attempts in salso procedure.
`...`	parameter of the generic method.

Details

put details here

Value

The function returns a vector with the cluster assignment of each observation.

References

D. B. Dahl, D. J. Johnson, and P. Müller (2022), Search Algorithms and Loss Functions for Bayesian Clustering, *Journal of Computational and Graphical Statistics*, 31(4), 1189-1201, doi:10.1080/10618600.2022.2069779.

Examples

```
data_vec <- as.numeric(c(rnorm(50,0,0.1), rnorm(50,1,0.25)))


out <- detect_cp(data = data_vec, n_iterations = 1000, n_burnin = 100,
                params = list(q = 0.25, phi = 0.1, a = 1, b = 1, c = 0.1), kernel = "ts")

posterior_estimate(out)
```

`print.ClustCpObj` *ClustCpObj print method*

Description

The `ClustCpObj` method prints which algorithm was run.

Usage

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

Arguments

`x`	an object of class `ClustCpObj`.
`...`	parameter of the generic method.

Examples

```
data_mat <- matrix(NA, nrow = 5, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_mat[4,] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
data_mat[5,] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp(data = data_mat, n_iterations = 5000, n_burnin = 1000,
                params = list(L = 1, B = 1000, phi = 0.5), kernel = "ts")

print(out)
```

print.DetectCpObj *DetectCpObj print method*

Description

The DetectCpObj method prints which algorithm was run.

Usage

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

Arguments

x	an object of class DetectCpObj.
...	parameter of the generic method.

Examples

```
data_mat <- matrix(NA, nrow = 3, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))

out <- detect_cp(data = data_mat, n_iterations = 2500, n_burnin = 500,
                 params = list(q = 0.25, k_0 = 0.25, nu_0 = 4,
                               S_0 = diag(1,3,3), m_0 = rep(0,3),
                               par_theta_c = 2, par_theta_d = 0.2,
                               prior_var_phi = 0.1), kernel = "ts")
print(out)
```

sim_epi_data *Simulate epidemiological data*

Description

Simulate epidemiological data

Usage

```
sim_epi_data(S0, I0, max_time, beta_vec, xi_0, user_seed = 1234L)
```

Arguments

S0	number of individuals in the population.
I0	number of infected individuals at time 0.
max_time	maximum observed time.
beta_vec	vector with the infection rate for each discrete time.
xi_0	the recovery rate of the population, must be in $(0, 1)$.
user_seed	seed for random distribution generation.

Value

Function sim_epi_data returns a vector with the simulated infection times.

Examples

```
betas <- c(rep(0.45, 25),rep(0.14,25))

inf_times <- as.numeric()

inf_times <- sim_epi_data(10000, 10, 50, betas, 1/8)
```

---

summary.ClustCpObj *ClustCpObj summary method*

Description

The ClustCpObj method returns a summary of the algorithm.

Usage

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

Arguments

object	an object of class ClustCpObj.
...	parameter of the generic method.

Examples

```
data_mat <- matrix(NA, nrow = 5, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))
data_mat[4,] <- as.numeric(c(rnorm(25,0,0.135), rnorm(75,1,0.225)))
```

```
data_mat[5,] <- as.numeric(c(rnorm(25,0,0.155), rnorm(75,1,0.280)))

out <- clust_cp(data = data_mat, n_iterations = 5000, n_burnin = 1000,
                params = list(L = 1, B = 1000, phi = 0.5), kernel = "ts")

summary(out)
```

---

summary.DetectCpObj *DetectCpObj summary method*

Description

The DetectCpObj method returns a summary of the algorithm.

Usage

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

Arguments

object	an object of class DetectCpObj;
...	parameter of the generic method.

Examples

```
data_mat <- matrix(NA, nrow = 3, ncol = 100)

data_mat[1,] <- as.numeric(c(rnorm(50,0,0.100), rnorm(50,1,0.250)))
data_mat[2,] <- as.numeric(c(rnorm(50,0,0.125), rnorm(50,1,0.225)))
data_mat[3,] <- as.numeric(c(rnorm(50,0,0.175), rnorm(50,1,0.280)))

out <- detect_cp(data = data_mat, n_iterations = 2500, n_burnin = 500,
                params = list(q = 0.25, k_0 = 0.25, nu_0 = 4,
                              S_0 = diag(1,3,3), m_0 = rep(0,3),
                              par_theta_c = 2, par_theta_d = 0.2,
                              prior_var_phi = 0.1), kernel = "ts")
summary(out)
```

Index