Package 'BayesChange'

February 15, 2025

Title Bayesian Methods for Change Points Analysis

Version 1.1.1

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

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 2025-02-14 23:00:15 UTC

Contents

`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,
  gamma = 1/8,
  alpha = 1,
  q = 0.1,
  dt = 0.1,
  a0 = 4,
  b0 = 10,
  c0 = 1,
  d0 = 1,
  MH_var = 0.01,
  S0 = 1,
  R0 = 0,
  p = 0.003,
  coars = 1,
  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.
gamma	recovery rate fixed constant for each population at each time.
alpha	α for the acceptance ration in the split-merge procedure.
q	probability of performing a split when updating the single order for the proposal procedure.
dt, a0, b0, c0, d0	parameters for the computation of the integrated likelihood of the survival functions.
MH_var	variance for the Metropolis-Hastings estimation of the proportion of infected at time 0.
S0, R0	parameters for the SDE solver.
p	prior average number of change points for each order.
coars	coarsening parameter.
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)

get_clust_VI(out$clust[1000:3000,])
```

`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,
  gamma,
  k_0,
  nu_0,
  phi_0,
  m_0,
  q = 0.5,
  alpha_SM = 0.1,
  coars = 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.
`gamma`, `k_0`, `nu_0`, `phi_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 split-merge proposal and acceleration step.
`coars`	coarsening coefficient, must be in (0,1].
`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,
                  gamma = 0.1, k_0 = 0.25, nu_0 = 5, phi_0 = diag(0.1,3,3), m_0 = rep(0,3))

get_clust_VI(out$clust[1000:3000,])
```

`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,
  gamma,
  a = 1,
  b = 1,
  c = 1,
  q = 0.5,
  alpha_SM = 0.1,
  coars = 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.
`gamma, a, b, c`	parameters γ of the integrated likelihood.
`q`	probability of a split in the split-merge proposal and acceleration step.
`alpha_SM`	α for the split-merge proposal and acceleration step.
`coars`	coarsening coefficient, must be in (0,1].
`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, gamma = 0.5)

get_clust_VI(out$clust[2500:5000,])
```

---

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,
  phi_0,
  m_0,
  par_theta_c = 1,
  par_theta_d = 1,
  prior_var_gamma = 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`, `phi_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_gamma`	parameters for the Gamma prior for γ.
`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:

- `$orders` a matrix where each row corresponds to the output order of the corresponding iteration.
- `time` computational time in seconds.
- `$gamma_MCMC` traceplot for γ.
- `$gamma_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.

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, phi_0 = diag(1,3,3), m_0 = rep(0,3),
                               par_theta_c = 2, par_theta_d = 0.2, prior_var_gamma = 0.1)

get_clust_VI(out$order)
```

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,
  phi,
  a,
  b,
  c,
  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.
phi	parameter ϕ of the integrated likelihood function.
a, b, c	parameters of the Normal-Gamma prior for μ and λ.
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,
                             phi = 0.1, a = 1, b = 1, c = 0.1)

get_clust_VI(out$order)
```

---

get_clust_VI *Estimate order*

Description

Estimate order

Usage

```
get_clust_VI(orders_mat)
```

Arguments

orders_mat	A matrix where each row corresponds to the output cluster of the corresponding iteration.

Value

Function get_clust_VI returns a point estimate for the clustering of the data.

---

psm *Compute the posterior similarity matrix*

Description

Compute the posterior similarity matrix

Usage

```
psm(M)
```

Arguments

M	A matrix where each row corresponds to the output cluster of the corresponding iteration.

Value

Function psm returns an $n \times n$ posterior similarity matrix.

`sim_epi_data` *Simulate epidemiological data*

Description

Simulate epidemiological data

Usage

```
sim_epi_data(S0, I0, max_time, beta_vec, gamma_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.
`gamma_0`	the recovery rate. for 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.

Index