Package: BayesChange (via r-universe)

October 24, 2026

Title Bayesian Methods for Change Point Analysis

Version 2.3.0

Description Performs change point detection on univariate and multivariate time series (Martínez & Mena, 2014, <doi:10.1214/14-BA878> ; Corradin, Danese & Ongaro, 2022, <doi:10.1016/j.ijar.2021.12.019>) and clusters time-dependent data with common change points (Corradin, Danese, KhudaBukhsh & Ongaro, 2026, <doi:10.1007/s11222-025-10756-x>).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppArmadillo, RcppGSL

Imports Rcpp, salso, dplyr, tidyr, ggplot2, ggpubr, coda, rlang, reshape2

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

LazyData true

NeedsCompilation yes

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

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

Repository https://cran.r-universe.dev

Date/Publication 2026-03-06 16:50:14 UTC

RemoteUrl https://github.com/cran/BayesChange

RemoteRef HEAD

RemoteSha 13e98cf12771205661f0fff6d55ec90812da5a19

Contents

clust_cp . . . 2
ClustCpObj . . . 5
detect_cp . . . 6
DetectCpObj . . . 9
epi_synthetic . . . 10
epi_synthetic_multi . . . 11
eu_inflation . . . 11
plot.ClustCpObj . . . 12
plot.DetectCpObj . . . 13
plot_psm.ClustCpObj . . . 14
posterior_estimate.ClustCpObj . . . 15
posterior_estimate.DetectCpObj . . . 16
print.ClustCpObj . . . 17
print.DetectCpObj . . . 18
sim_epi_data . . . 19
stock_multi . . . 20
stock_uni . . . 20
summary.ClustCpObj . . . 21
summary.DetectCpObj . . . 22

Index **23**

`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 epidemic diffusions.

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 epidemic diffusions: • M number of Monte Carlo iterations when computing the likelihood of the epidemic diffusion. • xi recovery rate fixed constant for each population at each time. • a0, b0 parameters for the computation of the integrated likelihood of the epidemic diffusions. • I0_var variance for the Metropolis-Hastings estimation of the proportion of infected at time 0. • avg_blk 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 epidemic diffusions.
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 containing the data.
- $n_iterations Total number of MCMC iterations.
- $n_burnin Number of burn-in iterations.
- $clust A matrix where each row corresponds to the cluster assignment from each iteration.

- `$orders` A multidimensional array where each slice is a matrix representing the latent order at each iteration.
- `$time` Total computational time (in seconds).
- `$entropy_MCMC` A `coda::mcmc` object containing the MCMC samples of the entropy.
- `$lkl` A `coda::mcmc` object containing the log-likelihood evaluated at each iteration.
- `$norm_vec` A vector containing the normalization constants computed at the beginning of the algorithm.
- `$I0_MCMC` A `coda::mcmc` object containing the MCMC trace of the initial infection proportion I_0.
- `$kernel_ts` TRUE if the kernel used corresponds to time series data.
- `$kernel_epi` TRUE if the kernel used corresponds to epidemic diffusion data.
- `$univariate_ts` TRUE if the data represent a univariate time series, FALSE if multivariate.

References

Corradin, R., Danese, L., KhudaBukhsh, W. R., & Ongaro, A. (2026). Model-based clustering of time-dependent observations with common structural changes. *Statistics and Computing*. doi:10.1007/s1122202510756x

Examples

```
## Univariate time series

data("stock_uni")

params_uni <- list(a = 1,
                   b = 1,
                   c = 1,
                   phi = 0.1)

out <- clust_cp(data = stock_uni[1:5,], n_iterations = 2000, n_burnin = 500,
                L = 1, q = 0.5, B = 1000, params = params_uni, kernel = "ts")

print(out)

## Multivariate time series

data("stock_multi")

params_multi <- list(m_0 = rep(0,2),
                     k_0 = 1,
                     nu_0 = 10,
                     S_0 = diag(1,2,2),
                     phi = 0.1)

out <- clust_cp(data = stock_multi[,,1:5], n_iterations = 2000, n_burnin = 500,
                L = 1, B = 1000, params = params_multi, kernel = "ts")
```

```
print(out)

## Epidemic diffusions

data("epi_synthetic_multi")

params_epi <- list(M = 100, xi = 1/8,
                   alpha_SM = 1,
                   a0 = 4,
                   b0 = 10,
                   I0_var = 0.1,
                   avg_blk = 2)

out <- clust_cp(epi_synthetic_multi, n_iterations = 2000, n_burnin = 500,
                L = 1, B = 1000, params = params_epi, kernel = "epi")

print(out)
```

`ClustCpObj` *ClustCpObj class constructor*

Description

A constructor for the `ClustCpObj` class, which stores the output of the change point detection and clustering algorithms.

Usage

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

Arguments

data	A vector or matrix containing the observed data.
n_iterations	Total number of MCMC iterations.
n_burnin	Number of burn-in iterations removed from posterior summaries.
clust	A matrix where each row contains the cluster assignments for one iteration.
orders	A multidimensional array where each slice is a matrix representing the latent order at each iteration.
time	Total computational time (in seconds).
norm_vec	A vector containing precomputed normalization constants.
entropy_MCMC	A coda::mcmc object containing the MCMC samples of the entropy.
lkl_MCMC	A coda::mcmc object containing the log-likelihood values at each iteration.
I0_MCMC	A coda::mcmc object with the MCMC trace of the initial infection proportion I_0.
kernel_ts	Logical; TRUE if the kernel corresponds to time-series data.
kernel_epi	Logical; TRUE if the kernel corresponds to epidemic diffusion data.
univariate_ts	Logical; TRUE if the data represent a univariate time series, FALSE for multivariate time series.

Value

An object of class ClustCpObj.

---

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`	if kernel = "ts" a vector or a matrix. If kernel = "epi" a matrix.
`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 δ. • `prior_delta_c` parameter of the shifted Gamma prior of δ. • `prior_delta_d` parameter of the shifted Gamma prior of δ. 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 $N(0, \sigma^2_\phi)$ posterior estimate of δ. • `prior_delta_c` parameter of the shifted Gamma prior of δ. • `prior_delta_d` parameter of the shifted Gamma prior of δ. If data are epidemic diffusions: • `M` number of Monte Carlo iterations when computing the likelihood of the epidemic diffusion. • `xi` recovery rate fixed constant for each population at each time. • `a0`,`b0` parameters for the computation of the integrated likelihood of the epidemic diffusions. • `I0_var` variance for the Metropolis-Hastings estimation of the proportion of infected at time 0.
`kernel`	can be "ts" if data are time series or "epi" if data are epidemic diffusions.
`print_progress`	If TRUE (default) print the progress bar.
`user_seed`	seed for random distribution generation.

Value

A `DetectCpObj` object containing:

- `$data` Vector or matrix with the input data.
- `$n_iterations` Total number of MCMC iterations.
- `$n_burnin` Number of burn-in iterations removed from posterior summaries.
- `$orders` Matrix of change–point allocations. Each row corresponds to an iteration; columns correspond to time indices.

- `$time` Computational time (in seconds).
- `$entropy_MCMC` A `coda::mcmc` object containing the MCMC samples of the entropy measure.
- `$lkl_MCMC` A `coda::mcmc` object containing the MCMC samples of the log-likelihood.
- `$phi_MCMC` A `coda::mcmc` object storing MCMC draws of γ.
- `$sigma_MCMC` A `coda::mcmc` object storing MCMC draws of σ.
- `$delta_MCMC` A `coda::mcmc` object storing MCMC draws of δ.
- `$I0_MCMC` A `coda::mcmc` object storing MCMC draws of I_0.
- `$kernel_ts` Logical; TRUE if the data are time series.
- `$kernel_epi` Logical; TRUE if the data follow an epidemic diffusion process.
- `$univariate_ts` Logical; TRUE if the time series is univariate, FALSE for multivariate.

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

Examples

```
## Univariate time series

data("eu_inflation")

params_uni <- list(a = 1, b = 1, c = 1, prior_var_phi = 0.1,
                   prior_delta_c = 1, prior_delta_d = 1)

out <- detect_cp(data = eu_inflation[1,], n_iterations = 2000,
                 n_burnin = 250, q = 0.5, params = params_uni,
                 kernel = "ts")

print(out)

## Multivariate time series

data("eu_inflation")

params_multi <- list(m_0 = rep(0,3),
                     k_0 = 1,
                     nu_0 = 10,
                     S_0 = diag(0.1,3,3),
                     prior_var_phi = 0.1,
                     prior_delta_c = 1,
                     prior_delta_d = 1)

out <- detect_cp(data = eu_inflation[1:3,], n_iterations = 2000,
                 n_burnin = 250, q = 0.5, params = params_multi, kernel = "ts")
```

```
print(out)


## Epidemic diffusions

data("epi_synthetic")

params_epi <- list(M = 100, xi = 1/8, a0 = 4, b0 = 10, I0_var = 0.1)

out <- detect_cp(data = epi_synthetic, n_iterations = 2000, n_burnin = 250,
                 q = 0.25, params = params_epi, kernel = "epi")

print(out)
```

`DetectCpObj` *DetectCpObj class constructor*

Description

Constructor for the `DetectCpObj` class. This class stores the output of the Bayesian change–point detection algorithm, including MCMC traces, allocation orders, and computational information.

Usage

```
DetectCpObj(
  data = NULL,
  n_iterations = NULL,
  n_burnin = NULL,
  orders = NULL,
  time = NULL,
  entropy_MCMC = NULL,
  lkl_MCMC = NULL,
  phi_MCMC = NULL,
  sigma_MCMC = NULL,
  delta_MCMC = NULL,
  I0_MCMC = NULL,
  kernel_ts = NULL,
  kernel_epi = NULL,
  univariate_ts = NULL
)
```

Arguments

`data`	A vector or matrix containing the observed time series.
`n_iterations`	Total number of MCMC iterations.

n_burnin	Number of burn-in iterations to discard.
orders	A matrix where each row corresponds to the latent block assignment (order) of the time indices at each MCMC iteration.
time	Computational time in seconds.
entropy_MCMC	A coda::mcmc object containing MCMC samples of the entropy measure.
lkl_MCMC	A coda::mcmc object containing MCMC samples of the log-likelihood.
phi_MCMC	A coda::mcmc object containing MCMC draws for γ.
sigma_MCMC	A coda::mcmc object containing MCMC draws for σ.
delta_MCMC	A coda::mcmc object containing MCMC draws for δ.
I0_MCMC	A coda::mcmc object containing MCMC draws for I_0.
kernel_ts	Logical; TRUE if the model for time series data is used.
kernel_epi	Logical; TRUE if the epidemic diffusion model is used.
univariate_ts	Logical; TRUE if the time series is univariate, FALSE otherwise.

`epi_synthetic` *Synthetic Epidemiological Time-Series Data*

Description

A toy dataset generated from a stochastic SIR-type epidemic model using the Doob–Gillespie algorithm. The transmission rate `beta` changes once over time, resultiing in a single change-point in the infection counts.

Usage

```
data(epi_synthetic)
```

Format

A 200 × 1 numeric matrix containing the daily number of infection events.

Details

The simulation follows the stochastic simulation framework of: Anderson, D. F. and Kurtz, T. G. (2015). *Stochastic Analysis of Biochemical Systems*. Springer International Publishing.

The simulation uses:

- `S0` = `10000`, `I0` = `50`
- `max_time` = `200`
- A piecewise-constant transmission rate vector with a change at time 130
- Infection event times aggregated using `floor()`

epi_synthetic_multi *Multivariate Synthetic Epidemiological Time-Series Data*

Description

A multivariate synthetic infection-count dataset generated from three independent stochastic epidemic processes simulated using the Doob–Gillespie algorithm. Each time series has a different transmission rate, resulting in distinct change point structures.

Usage

```
data(epi_synthetic_multi)
```

Format

A 3×200 numeric matrix. Each row represents one synthetic epidemic time series and each column corresponds to a discrete time point.

Details

The simulation follows the stochastic framework described in: Anderson, D. F. and Kurtz, T. G. (2015). *Stochastic Analysis of Biochemical Systems*. Springer International Publishing.

All three epidemic processes use:

- `S0` = `100000`, `I0` = `20`
- `max_time` = `200`
- `xi_0` = `1/8`

The three `beta` vectors differ in their change-point locations and values:

- Series 1: `0.211` → `0.55` at time 120
- Series 2: `0.215` → `0.52` at time 120
- Series 3: `0.193` → `0.53` at time 30

eu_inflation *EU Inflation Dataset (Standardized Matrix Form)*

Description

This dataset contains standardized monthly inflation rates (annual average rate of change) in the Harmonized Index of Consumer Prices (HICP) for the European Union. Data cover February 1997 to December 2024 and are organized across the 12 COICOP expenditure categories.

Usage

```
data(eu_inflation)
```

Format

A numeric matrix with 12 rows and 355 columns:

Rows COICOP categories

Columns Monthly inflation observations (Feb 1997–Dec 2024)

Details

Each row corresponds to one COICOP category, and each column corresponds to a monthly observation.

Source

Eurostat — *HICP (prc_hicp_aind)* available at the Eurostat Data Explorer.

---

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 epidemic diffusions colored according to the final partition.

Examples

```
data("stock_uni")

params_uni <- list(a = 1,
                   b = 1,
                   c = 1,
                   phi = 0.1)

out <- clust_cp(data = stock_uni[1:3,], n_iterations = 1000, n_burnin = 100,
                L = 1, q = 0.5, B = 500, params = params_uni, kernel = "ts")

plot(out)
```

---

plot.DetectCpObj *Plot estimated change points*

Description

The `plot` method plots the estimated 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`	Logical; 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

```
## Univariate time series

data("eu_inflation")

params_uni <- list(a = 1, b = 1, c = 1, prior_var_phi = 0.1,
                   prior_delta_c = 1, prior_delta_d = 1)

out <- detect_cp(data = eu_inflation[1,], n_iterations = 1000,
                 n_burnin = 100, q = 0.5, params = params_uni,
                 kernel = "ts")

plot(out)
```

---

plot_psm.ClustCpObj *Plot the Posterior Similarity Matrix (PSM) for a ClustCpObj*

Description

This function computes and visualizes the posterior similarity matrix (PSM) from a ClustCpObj object. The PSM shows the posterior co-clustering probabilities of all observations.

Usage

```
## S3 method for class 'ClustCpObj'
plot_psm(object, reorder = TRUE, title = "Posterior Similarity Matrix", ...)
```

Arguments

object	an object of class ClustCpObj.
reorder	Logical; if TRUE (default), items are reordered using hierarchical clustering to highlight clusters in the final plot
title	Character; the plot title (default: "Posterior Similarity Matrix").
...	parameter of the generic method.

Value

A ggplot2 object representing the posterior similarity matrix.

Examples

```
data("stock_uni")

params_uni <- list(a = 1,
                   b = 1,
                   c = 1,
                   phi = 0.1)

out <- clust_cp(data = stock_uni[1:3,], n_iterations = 1000, n_burnin = 100,
                L = 1, q = 0.5, B = 500, params = params_uni, kernel = "ts")
plot_psm(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.

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("stock_uni")

params_uni <- list(a = 1,
                   b = 1,
                   c = 1,
                   phi = 0.1)

out <- clust_cp(data = stock_uni[1:3,], n_iterations = 1000, n_burnin = 100,
                L = 1, q = 0.5, B = 500, params = params_uni, 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,
  show_cp = FALSE,
  loss = "VI",
  maxNClusters = 0,
  nRuns = 16,
  maxZealousAttempts = 10,
  ...
)
```

Arguments

object	An object of class DetectCpObj.
show_cp	Logical; if TRUE show change point locations; FALSE show cluster assignment.
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 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("eu_inflation")

params_uni <- list(a = 1, b = 1, c = 1, prior_var_phi = 0.1,
                   prior_delta_c = 1, prior_delta_d = 1)

out <- detect_cp(data = eu_inflation[1,], n_iterations = 1000,
                 n_burnin = 100, q = 0.5, params = params_uni,
                 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("stock_uni")

params_uni <- list(a = 1,
                   b = 1,
                   c = 1,
                   phi = 0.1)

out <- clust_cp(data = stock_uni[1:3,], n_iterations = 1000, n_burnin = 100,
                L = 1, q = 0.5, B = 500, params = params_uni, 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("eu_inflation")

params_uni <- list(a = 1, b = 1, c = 1, prior_var_phi = 0.1,
                   prior_delta_c = 1, prior_delta_d = 1)

out <- detect_cp(data = eu_inflation[1,], n_iterations = 1000,
                 n_burnin = 100, q = 0.5, params = params_uni,
                 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)
```

stock_multi *Daily Stock Price Dataset (Multivariate Open/Close Series)*

Description

This dataset contains detrended and standardized daily opening and closing stock prices for the 50 largest companies in the S&P 500 index. Data cover the period from January 1, 2020 to January 1, 2022, with 505 daily observations per company.

Usage

```
data(stock_multi)
```

Format

A numeric array of dimension `2 × 505 × 50`:

Dimension 1 Price type: `"Open"`, `"Close"`

Dimension 2 Daily observations (505)

Dimension 3 Companies (S&P 500 top 50)

Source

Yahoo Finance (https://finance.yahoo.com/)

stock_uni *Daily Stock Price Dataset (Univariate Mean Series)*

Description

This dataset contains detrended and standardized daily mean stock prices for the 50 largest companies (by market capitalization) in the S&P 500 index. Data cover the period from January 1, 2020 to January 1, 2022.

Usage

```
data(stock_uni)
```

Format

A numeric matrix with 50 rows and 505 columns:

Rows Companies (S&P 500 top 50)

Columns Daily mean prices (505 observations)

Details

For each company, the mean price is computed as the average between the daily high and low prices. Each resulting time series contains 505 daily observations.

Source

Yahoo Finance (https://finance.yahoo.com/)

---

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("stock_uni")

params_uni <- list(a = 1,
                   b = 1,
                   c = 1,
                   phi = 0.1)

out <- clust_cp(data = stock_uni[1:3,], n_iterations = 1000, n_burnin = 100,
                L = 1, q = 0.5, B = 500, params = params_uni, 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("eu_inflation")

params_uni <- list(a = 1, b = 1, c = 1, prior_var_phi = 0.1,
                   prior_delta_c = 1, prior_delta_d = 1)

out <- detect_cp(data = eu_inflation[1,], n_iterations = 1000,
                 n_burnin = 100, q = 0.5, params = params_uni,
                 kernel = "ts")
summary(out)
```

Index

∗ **datasets**
epi_synthetic, 10
epi_synthetic_multi, 11
eu_inflation, 11
stock_multi, 20
stock_uni, 20

clust_cp, 2
ClustCpObj, 5

detect_cp, 6
DetectCpObj, 9

epi_synthetic, 10
epi_synthetic_multi, 11
eu_inflation, 11

plot.ClustCpObj, 12
plot.DetectCpObj, 13
plot_psm.ClustCpObj, 14
posterior_estimate.ClustCpObj, 15
posterior_estimate.DetectCpObj, 16
print.ClustCpObj, 17
print.DetectCpObj, 18

sim_epi_data, 19
stock_multi, 20
stock_uni, 20
summary.ClustCpObj, 21
summary.DetectCpObj, 22