Package ‘changepointGA’

March 28, 2026

Type Package

Title Changepoint Detection via Modified Genetic Algorithm

Version 0.1.4

Date 2026-03-23

Description The Genetic Algorithm (GA) is used to perform changepoint analysis in time series data. The package also includes an extended island version of GA, as described in Lu, Lund, and Lee (2010, <doi:10.1214/09-AOAS289>). By mimicking the principles of natural selection and evolution, GA provides a powerful stochastic search technique for solving combinatorial optimization problems. In 'changepointGA', each chromosome represents a changepoint configuration, including the number and locations of changepoints, hyperparameters, and model parameters. The package employs genetic operators—selection, crossover, and mutation—to iteratively improve solutions based on the given fitness (objective) function. Key features of 'changepointGA' include encoding changepoint configurations in an integer format, enabling dynamic and simultaneous estimation of model hyperparameters, changepoint configurations, and associated parameters. The detailed algorithmic implementation can be found in the package vignettes and in the paper of Li (2024, <doi:10.48550/arXiv.2410.15571>).

License Apache License (>= 2.0)

RoxygenNote 7.3.3

Depends R (>= 4.3.0)

Imports methods, foreach, doParallel, Rcpp, clue, stats, utils

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++17

URL `https://github.com/mli171/changepointGA`

BugReports `https://github.com/mli171/changepointGA/issues`

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Mo Li [aut, cre],
QiQi Lu [aut]

Maintainer Mo Li <mo.li@louisiana.edu>

Repository CRAN

Date/Publication 2026-03-28 15:30:08 UTC

Contents

`amoc_crossover` *Average crossover operator to produce offspring for AMOC problem*

Description

In this crossover operator designed for AMOC problem, the new child is produced by taking the average of the changepoint locations from dad and `mom` and round to an integer. Note, every chromosome has at most one candidate changepoint location.

Usage

```
amoc_crossover(mom, dad, prange = NULL, minDist, lmax, N)
```

Arguments

`mom`	Among two selected individuals, `mom` represents the selected chromosome representation with lower fitness function value.
`dad`	Among two selected individuals, `dad` represents the selected chromosome representation with larger fitness function value.
`prange`	The default value is `NULL`. If there is no requirement on model order selection, such an auxiliary argument is needed for `GA` and `IslandGA` functions.
`minDist`	The minimum length between two adjacent changepoints.
`lmax`	The maximum possible length of the chromosome representation.
`N`	The length of time series.

Value

The child chromosome that produced from `mom` and `dad` for next generation.

`amoc_mutation` *Jump mutation operator to produce offspring for AMOC problem*

Description

In a certain probability, the `mutation` genetic operator can be applied to generate a new `child`. In this AMOC mutation operator, the new child changepoint location can be down via a "jump" method. The child changepoint location will jump `minDist` time units either to the left or right to produce the changepoint location for the mutated child. The jump direction is randomly decided with 0.5 probability.

Usage

```
amoc_mutation(
  child,
  prange = NULL,
  minDist,
  pchangepoint = NULL,
  lmax = NULL,
  mmax = NULL,
  N = NULL
)
```

Arguments

`child`	The child chromosome resulting from the `crossover` genetic operator.
`prange`	The default value is `NULL`. If there is no requirement on model order selection, such an auxiliary argument is needed for `GA` and `IslandGA` functions.
`minDist`	The minimum length between two adjacent changepoints in `amoc_selection` operator, which is also the jump magnitude in the `amoc_mutation` operator.

`pchangepoint`	An auxiliary argument is needed for `GA` and `IslandGA` functions.
`lmax`	An auxiliary argument is needed for `GA` and `IslandGA` functions.
`mmax`	An auxiliary argument is needed for `GA` and `IslandGA` functions.
`N`	An auxiliary argument is needed for `GA` and `IslandGA` functions.

Value

The resulting child chromosome representation.

`amoc_population` *Random population initialization for AMOC problem*

Description

Randomly generate the individuals' chromosomes (changepoint configurations) to construct the first generation population for the at most one changepoint (AMOC) problem.

Usage

```
amoc_population(popSize, prange, N, minDist, pchangepoint, mmax, lmax)
```

Arguments

`popSize`	An integer represents the number of individual in each population for GA (or subpopulation for IslandGA).
`prange`	Default is `NULL` for only changepoint detection. If `prange` is specified as a list object, which contains the range of each model order parameters for order selection (integers). The number of order parameters must be equal to the length of `prange`.
`N`	The length of time series.
`minDist`	The minimum length between two adjacent changepoints.
`pchangepoint`	The probability that a changepoint can occur.
`mmax`	The maximum possible number of changepoints in the data set.
`lmax`	The maximum possible length of the chromosome representation.

Details

Given the possible candidate changepoint location set, each chromosome in the first generation population can be obtained by randomly sampling one location from the candidate set. The first element of every chromosome represent the number of changepoints and the last non-zero element always equal to the length of time series plus one (N+1).

Value

A matrix that contains each individual's chromosome.

amoc_selection *The parents selection genetic algorithm operator for AMOC problem*

Description

The genetic algorithm require to select a pair of chromosomes, representing `dad` and `mom`, for the `crossover` operator to produce offspring (individual for next generation). Here, the same linear ranking method in `selection_linear_rank` is used to select a pair of chromosomes for `dad` and `mom` in the at most one changepoint (AMOC) problem. By default, the dad has better fit/smaller fitness function value/larger rank than `mom`.

Usage

```
amoc_selection(pop, popFit)
```

Arguments

`pop`	A matrix contains the chromosomes for all individuals. The number of rows is equal to `lmax` and the number of columns is equal to the `popSize`.
`popFit`	A vector contains the objective function value (population fit) being associated to each individual chromosome from above.

Value

A list contains the chromosomes for `dad` and `mom`.

arima_bic *Example function: Calculating BIC for AR(1) model*

Description

The objective function for changepoint search in Autoregressive moving average with model order selection via Bayesian Information Criterion (BIC).

Usage

```
arima_bic(chromosome, plen = 0, XMat, Xt)
```

Arguments

`chromosome`	The chromosome consists of the number of changepoints, the order of AR part, the order of MA part, the changepoint locations, and a value of time series length plus 1 (N+1) indicating the end of the chromosome.
`plen`	The number of model order parameters that need to be selected. If model order selection needs to be performed simultaneously with the changepoint detection task, `plen` should be nonzero.

XMat	A matrix contains the covariates, but not includes changepoint effects, for time series regression.
Xt	The simulated ARMA time series from `ts.sim` function.

Value

The BIC value of the objective function.

Examples

```
Ts <- 1000
betaT <- c(0.5) # intercept
XMatT <- matrix(1, nrow = Ts, ncol = 1)
colnames(XMatT) <- "intercept"
sigmaT <- 1
phiT <- c(0.5)
thetaT <- NULL
DeltaT <- c(2, -2)
Cp.prop <- c(1 / 4, 3 / 4)
CpLocT <- floor(Ts * Cp.prop)

myts <- ts_sim(
  Ts = Ts, beta = betaT, XMat = XMatT, sigma = sigmaT, phi = phiT, theta = thetaT,
  Delta = DeltaT, CpLoc = CpLocT, seed = 1234
)

# candidate changepoint configuration
chromosome <- c(2, 250, 750, 1001)
arima_bic(chromosome, XMat = XMatT, Xt = myts)
```

arima_bic_order_pdq *Calculate BIC for changepoint detection with ARIMA order selection*

Description

Objective function for simultaneous selection of changepoint locations and ARIMA model orders using the Bayesian Information Criterion (BIC).

Usage

```
arima_bic_order_pdq(chromosome, plen = 3, XMat, Xt)
```

Arguments

chromosome	A numeric vector encoding the candidate model. The chromosome is assumed to have the form `c(m, p, d, q, tau1, tau2, ..., tauK)`, where `m` is the number of changepoints, `p`, `d`, and `q` are the ARIMA orders, and the remaining entries are candidate changepoint locations.

`plen`	An integer giving the number of model-order entries in the chromosome. The default is `3`, corresponding to (p, d, q).
`XMat`	A matrix of regression covariates excluding changepoint indicators.
`Xt`	A numeric vector containing the observed time series.

Details

Each chromosome encodes the number of changepoints, the ARIMA orders (p, d, q), and candidate changepoint locations. For a given chromosome, the function constructs the corresponding changepoint design matrix, combines it with the user-supplied regression design matrix, fits an ARIMA model with regressors, and returns the BIC value.

The first element of `chromosome` specifies the number of changepoints. The next `plen` elements specify the ARIMA orders, which are rounded to integers. Remaining elements are treated as candidate changepoint locations.

If changepoints are present, the function constructs a block-indicator matrix whose columns represent successive changepoint segments. This matrix is then appended to `XMat` to form the regression design.

Before fitting, columns with zero variance or non-finite values are removed. Remaining regressors are mean-centered. The ARIMA model is then fitted using `arima` with `include.mean = FALSE`, and the corresponding BIC is returned.

If model fitting fails, the function returns a large penalty value (`1e10`) so that the candidate is disfavored during optimization.

Value

A numeric scalar giving the BIC of the fitted candidate model. If the ARIMA fit fails, the function returns `1e10`.

Examples

```
Ts <- 200
XMatT <- matrix(1, nrow = Ts, ncol = 1)
colnames(XMatT) <- "intercept"

Xt <- ts_sim(
  Ts = Ts, beta = 0.5, XMat = XMatT, sigma = 1,
  phi = 0.5, theta = 0.3, d = 1,
  Delta = c(2, -2), CpLoc = c(50, 150), seed = 1234
)

## chromosome = c(m, p, d, q, tau1, tau2, ...)
chromosome <- c(2, 1, 1, 1, 50, 150, Ts + 1)

arima_bic_order_pdq(
  chromosome = chromosome,
  plen = 3,
  XMat = XMatT,
  Xt = Xt
)
```

arima_bic_order_pq — *Calculating BIC for Multiple changepoint detection with AR and MA order selection*

Description

The objective function for changepoint search in Autoregressive moving average with model order selection via Bayesian Information Criterion (BIC).

Usage

```
arima_bic_order_pq(chromosome, plen = 2, XMat, Xt)
```

Arguments

`chromosome`	A vector consists of the number of changepoints, the order of AR component (refers to the number of lagged terms used to model the current value of a time series), the order of MA component (refers to the number of lagged error terms used to model the current value of a time series), the changepoint locations, and a value of time series sample size plus 1 ($N+1$) indicating the end of the chromosome.
`plen`	The number of model order parameters that need to be selected. If model order selection needs to be performed simultaneously with the changepoint detection task, `plen` should be nonzero.
`XMat`	A matrix contains the covariates, but not includes changepoint effects, for time series regression.
`Xt`	The simulated ARMA time series from `ts.sim` function.

Value

The BIC value of the objective function.

Examples

```
Ts <- 1000
XMatT <- matrix(1, nrow = Ts, ncol = 1)
Xt <- ts_sim(
  Ts = Ts, beta = 0.5, XMat = XMatT, sigma = 1, phi = 0.5, theta = 0.8,
  Delta = c(2, -2), CpLoc = c(250, 750), seed = 1234
)

# one chromosome representation
chromosome <- c(2, 1, 1, 250, 750, 1001)
arima_bic_order_pq(chromosome, plen = 2, XMat = XMatT, Xt = Xt)
```

`cptga` *Genetic algorithm*

Description

Perform the genetic algorithm for changepoint detection. This involves the minimization of an objective function using a genetic algorithm (GA). The algorithm can be run sequentially or with explicit parallelization.

Usage

```
cptga(
  ObjFunc,
  N,
  prange = NULL,
  popSize = 200,
  pcrossover = 0.95,
  pmutation = 0.3,
  pchangepoint = 0.01,
  minDist = 1,
  mmax = NULL,
  lmax = NULL,
  maxgen = 50000,
  maxconv = 5000,
  option = "cp",
  monitoring = FALSE,
  parallel = FALSE,
  nCore = NULL,
  tol = 1e-05,
  seed = NULL,
  popInitialize = "random_population",
  suggestions = NULL,
  selection = "selection_linear_rank",
  crossover = "uniform_crossover",
  mutation = "mutation",
  ...
)
```

Arguments

`ObjFunc`	The fitness function to be minimized. Users can specify any R or Rcpp function as the fitness function, setting the input as the potential solution to the optimization problem and returning a numerical value as the output/fitness. Depending on the user-specified chromosome representation, the optimization task can be changepoint detection only or changepoint detection plus model order selection, which can be specified via the `option` parameter. When `option="both"`, the list `prange` must be specified to give the range of model orders.

`N`	The sample size of the time series.
`prange`	The default value is `NULL` for changepoint detection only task. If both model order selection and changepoint detection are required, the `prange` argument should be provided as a list. Each element in this list must specify the range for a corresponding model order parameter, and the length of the list object of `prange` must match the number of order parameters to be estimated.
`popSize`	An integer represents the number of individuals/chromosomes in each population.
`pcrossover`	The probability that the crossover operator will apply to the two selected parents' chromosomes to produce the offspring. The typical value is close to 1, with the default setting in this package being 0.95.
`pmutation`	The probability that the mutation operator applies on one individual chromosome. Similar to the natural mutation process, new genetic information is introduced to the offspring chromosome with a relatively small probability (close to 0), with a default value of 0.3.
`pchangepoint`	The probability that a changepoint has occurred with a default value of 0.01. User could change this probability based on domain knowledge and the time series length. This probability is used during population initialization and in the creation of new chromosomes by the mutation operator. By default, the mutation operator function generates a new individual as the mutated offspring.
`minDist`	The minimum length between two adjacent changepoints. Default value equals to one.
`mmax`	The maximum number of changepoints allowed in the time series data corresponds to the maximum possible length of $\boldsymbol{\tau}$. For a time series of length 1000 and we only want to detect the changepoint (`option="cp"`), the default value is 499. The suggested value should be based on the length of the time series. For instance, if a time series has length N, the recommended `mmax` should be $N/2 - 1$. It is suggested to add the number of model hyperparameters if both changepoint detection and model order selection tasks are of-interested simultaneously (`option="both"`).
`lmax`	The maximum possible length of the chromosome representation. For a time series of length 1000 and we only want to detect the changepoint (`option="cp"`), the default value is 501. The suggested value should be based on the length of the time series. For instance, if a time series has length N, the recommended `lmax` should be $2 + N/2 - 1$. It is suggested to add the number of model hyperparameters if both changepoint detection and model order selection tasks are of-interested simultaneously (`option="both"`).
`maxgen`	The maximum number of generations the GA can run before the search is forcibly terminated.
`maxconv`	If the overall best fitted value doesn't change after `maxconv` consecutive migrations, the GA algorithm stops.
`option`	A character string controls the optimization task. `"cp"` indicates the task is changepoint detection only. `"both"` indicates the task will include both changepoint detection and model order selection.

`monitoring`	A logical value with either `TRUE` or `FALSE`, indicating whether to print out summarized results (current best fitness function value and its corresponding C) for each generation from the GA.
`parallel`	A logical value with either `TRUE` or `FALSE`, indicating whether to use multiple cores for parallel computation of the fitness function values for individuals after population initialization.
`nCore`	An integer with the default value of `NULL`. It represents the number of cores used in parallel computing. The value of `nCore` must be less than the number of physical cores available.
`tol`	An numerical value with the default value of `1e-05`. The tolerance level that helps evaluate whether the two iterations have the same fitness value, which aids in determining GA termination.
`seed`	An integer with the default value of `NULL`. An single integer allows function produce reproducible results.
`popInitialize`	A function or sourced function name character string. It should be designed for initializing a population. The default population initialization is random initialization with some imposed constraints. See `random_population` for example. The function returned object is a matrix, `pop`. The users can specified their own `population` function. It could also be a matrix object, which contain the user specified chromosome. By default, each column represents one individual chromosome. See `random_population` for details.
`suggestions`	A list object. Default value is `NULL`. Each element includes better or more reasonable guessed changepoints locations, which will resulting in one chromosome. If the number of provided suggestions equals to `popSize`, all the suggestions will be used as population. If the number of provided suggestions is less than `popSize`, the function from `popInitialize` will generate the remaining individual chromosomes. The number of provided suggestions cannot be greater than `popSize`. Having better `suggestions` can help GA converges faster.
`selection`	A function or sourced function name character string. This GA operator can help select `mom` and `dad` from current generation population, where `dad` is set to have better fit (smaller fitness function values). The default for selection uses the linear rank selection method. See `selection_linear_rank` for example. The function returned object is a list contain the chromosomes for `mom` and `dad`.
`crossover`	A function or sourced function name character string. This GA operator can apply crossover to the chosen parents to produce child for next generation with specified probability. The default for crossover uses the uniform crossover method. See `uniform_crossover` for details in the default crossover operator. The function returned object is a vector contain the chromosomes for `child`.
`mutation`	A function or sourced function name character string. This GA operator can apply mutation to the produced `child` with the specified probability `pmutation`. See `mutation` for details in the default mutation operator. The function returned object is a vector contain `child` chromosome representation.
`...`	additional arguments that will be passed to the fitness function.

Details

For any pre-specified time series model with a specified set of changepoint locations, model fit is evaluated using a fitness function $Q(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{s}, \boldsymbol{\tau}, \boldsymbol{\beta})'$ denotes the full parameter vector. Here, $\boldsymbol{s}$ denotes the set of model hyperparameters, potentially including the AR or MA orders, the degree of ARIMA differencing, or the periodic AR order for PAR models. The vector $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_m\}$ specifies the changepoint locations, with the number of changepoints m inferred as part of the estimation. Each individual chromosome representation is specified as a vector,

$$C = (m, \boldsymbol{s}, \boldsymbol{\tau}, N + 1)',$$

where m represents the number of changepoints and is also equivalent to the length of vector $\boldsymbol{\tau}$ containing all the changepoint locations. $\boldsymbol{s}$ contains the integer-valued parameter for time series model structure specification, such as AR, MA, or PAR orders. If no model order selection is desired, then $\boldsymbol{s}$ is omitted and the GA detects changepoints only. The changepoint locations in $\boldsymbol{\tau}$ are encoded as interger values between 2 and N, allowing the length of $\boldsymbol{\tau}$ to vary dynamically with m. The value $N + 1$ is appended at the end of C to serve as a delimiter marking the boundary of the chromosome.

Value

Return an object class `cptga-class`. See `cptga-class` for a more detailed description.

Examples

```
Ts <- 1000
XMatT <- matrix(1, nrow = Ts, ncol = 1)
Xt <- ts_sim(
  Ts = Ts, beta = 0.5, XMat = XMatT, sigma = 1, phi = 0.5, theta = NULL,
  Delta = c(2, -2), CpLoc = c(250, 750), seed = 1234
)

## Multiple changepoint detection without model order selection

# without suggestions
GAres <- cptga(ObjFunc = arima_bic, N = Ts, XMat = XMatT, Xt = Xt)
summary(GAres)
plot(GAres, data = Xt)

# with suggestions
suggestions <- list(NULL, 250, c(250, 500), c(250, 625), c(250, 500, 750))
GAres <- cptga(ObjFunc = arima_bic, N = Ts, suggestions = suggestions, XMat = XMatT, Xt = Xt)
summary(GAres)
plot(GAres, data = Xt)


## Multiple changepoint detection with model order selection

prange <- list(ar = c(0, 3), ma = c(0, 3))

# without suggestions
```

```
GAres <- cptga(
  ObjFunc = arima_bic_order_pq, N = Ts, prange = prange,
  option = "both", XMat = XMatT, Xt = Xt
)
summary(GAres)
plot(GAres, data = Xt)

# with suggestions
suggestions <- list(NULL, 250, c(250, 500), c(250, 625), c(250, 500, 750))
GAres <- cptga(
  ObjFunc = arima_bic_order_pq, N = Ts, prange = prange,
  suggestions = suggestions, option = "both", XMat = XMatT, Xt = Xt
)
summary(GAres)
plot(GAres, data = Xt)
```

`cptga-class` *S4 Class Definition for 'cptga'*

Description

S4 Class for Genetic Algorithm-Based Changepoint Detection

Usage

```
## S4 method for signature 'cptga'
summary(object, ...)
```

Arguments

`object`	An object of class `cptga`.
`...`	Additional arguments (ignored).

Details

An object of class `cptga` stores results and configuration settings for changepoint detection using a Genetic Algorithm (GA), optionally with simultaneous model order selection. This class records GA control parameters, intermediate population structures, and the optimal solution found.

Value

An object of class `cptga`.

Slots

- `call` The matched call that created the object.
- `N` The sample size of the time series.
- `prange` A list object. Default is `NULL`. If specified, it contains the ranges for each model order parameter (integers). Required when `option = "both"` is used for joint changepoint and model selection.
- `popSize` An integer representing the number of individuals in each GA population.
- `pcrossover` The probability that the crossover operator is applied to two chromosomes.
- `pmutation` The probability that the mutation operator is applied to a chromosome.
- `pchangepoint` The prior probability that a changepoint has occurred at each location.
- `minDist` The minimum allowed distance between two adjacent changepoints.
- `mmax` The maximum possible number of changepoints. Typically set based on time series length and `option`.
- `lmax` The maximum length of the chromosome. Typically set based on time series length and `option`.
- `maxgen` The maximum number of generations the GA is allowed to run.
- `maxconv` If the optimal fitness value does not improve over this many generations, GA stops.
- `option` A character string: either `"cp"` for changepoint detection only, or `"both"` for changepoint detection and model order selection.
- `monitoring` Logical. If `TRUE`, prints intermediate GA progress.
- `parallel` Logical. If `TRUE`, enables parallel computation for fitness evaluation.
- `nCore` Integer or `NULL`. Number of cores used for parallel computation when `parallel = TRUE`.
- `tol` Numeric. Tolerance for determining GA convergence. Default is `1e-5`.
- `seed` An integer or `NULL`. Random seed for reproducibility.
- `suggestions` A list or `NULL`. Each element provides suggested changepoint locations to guide initial population design and potentially accelerate convergence.
- `population` A matrix where each row represents an individual chromosome in the current population.
- `fitness` A numeric vector containing the fitness values of individuals in the current generation.
- `overbestchrom` A vector representing the best chromosome found over all generations.
- `overbestfit` A numeric scalar. The best (smallest) fitness value achieved.
- `bestfit` A numeric vector recording the best fitness value in each generation.
- `count` A numeric value indicating the number of generations the GA actually ran.
- `convg` A numeric vector representing convergence information. A value of `0` indicates the algorithm successful completion. A value of `1` indicates the the total number of generations exceeds the pre-specified `maxgen` limit.

See Also

`cptga`, `cptga-class`, `random_population`, `selection_linear_rank`, `uniform_crossover`, `mutation`.

cptgaisl *Island model based genetic algorithm*

Description

Perform the modified island-based genetic algorithm (IslandGA) for multiple changepoint detection. Minimization of an objective function using genetic algorithm (GA). The algorithm can be run sequentially or in explicit parallelisation.

Usage

```
cptgaisl(
  ObjFunc,
  N,
  prange = NULL,
  popSize = 200,
  numIslands = 5,
  pcrossover = 0.95,
  pmutation = 0.3,
  pchangepoint = 0.01,
  minDist = 1,
  mmax = NULL,
  lmax = NULL,
  maxMig = 1000,
  maxgen = 50,
  maxconv = 100,
  option = "cp",
  monitoring = FALSE,
  parallel = FALSE,
  nCore = NULL,
  tol = 1e-05,
  seed = NULL,
  popInitialize = "random_population",
  suggestions = NULL,
  selection = "selection_linear_rank",
  crossover = "uniform_crossover",
  mutation = "mutation",
  ...
)
```

Arguments

`ObjFunc` The fitness function to be minimized. Users can specify any R or Rcpp function as the fitness function, setting the input as the potential solution to the optimization problem and returning a numerical value as the output/fitness. Depending on the user-specified chromosome representation, the optimization task can be changepoint detection only or changepoint detection plus model order selection,

	which can be specified via the `option` parameter. When `option="both"`, the list `prange` must be specified to give the range of model orders.
`N`	The sample size of the time series.
`prange`	The default value is `NULL` for changepoint detection only task. If both model order selection and changepoint detection are required, the `prange` argument should be provided as a list. Each element in this list must specify the range for a corresponding model order parameter, and the length of the list object of `prange` must match the number of order parameters to be estimated.
`popSize`	An integer representing the total number of individuals in each generation, which equal to the number of islands multiplied by the size of each island (i.e., `popSize = numIslands × Islandsize`).
`numIslands`	An integer representing the number of islands (sub-populations).
`pcrossover`	The probability that the crossover operator will apply to the two selected parents' chromosomes to produce the offspring. The typical value is close to 1, with the default setting in this package being 0.95.
`pmutation`	The probability that the mutation operator applies on one individual chromosome. Similar to the natural mutation process, new genetic information is introduced to the offspring chromosome with a relatively small probability (close to 0), with a default value of 0.3.
`pchangepoint`	The probability that a changepoint has occurred. User could change this probability based on domain knowledge and the time series length. This probability is used during population initialization and in the creation of new chromosomes by the mutation operator. By default, the mutation operator function generates a new individual as the mutated offspring.
`minDist`	The minimum length between two adjacent changepoints. Default value equals to one.
`mmax`	The maximum number of changepoints allowed in the time series data corresponds to the maximum possible length of $\boldsymbol{\tau}$. For a time series of length 1000 and we only want to detect the changepoint (`option="cp"`), the default value is 499. The suggested value should be based on the length of the time series. For instance, if a time series has length N, the recommended `mmax` should be $N/2-1$. It is suggested to add the number of model hyperparameters if both changepoint detection and model order selection tasks are of-interested simultaneously (`option="both"`).
`lmax`	The maximum possible length of the chromosome representation. For a time series of length 1000 and we only want to detect the changepoint (`option="cp"`), the default value is 501. The suggested value should be based on the length of the time series. For instance, if a time series has length N, the recommended `lmax` should be $2+N/2-1$. It is suggested to add the number of model hyperparameters if both changepoint detection and model order selection tasks are of-interested simultaneously (`option="both"`).
`maxMig`	An integer indicates the maximum number of migrations. After conducting `maxMig` migrations, the island model GA stops.
`maxgen`	An integer indicates the maximum number of generations that each island (subpopulation) undergoes before migration. It also determines the requency of

	migration. The migration will occur after `maxgen` generations for each island (sub-population).
`maxconv`	An integer value is also used for algorithm termination. If the overall best-fitted value remains unchanged after `maxconv` consecutive migrations, the island model GA will terminate.
`option`	A character string controls the optimization task. `"cp"` indicates the task is changepoint detection only. `"both"` indicates the task will include both changepoint detection and model order selection.
`monitoring`	A logical value with either `TRUE` or `FALSE`, indicating whether to print out summarized results (current best fitness function value and its corresponding C) for each generation from the GA.
`parallel`	A logical value with `TRUE` or `FALSE`, indicating whether to use multiple cores for parallel computation. Different from the basic GA, in this approach, each island evolves independently. New population generation—including parent selection, crossover, and mutation—is processed independently and in parallel for each island, further saving computation time. Once the new population generation is complete, the migration operator is performed sequentially.
`nCore`	An integer indicates the number of cores utilized in parallel computing. For the island model GA, the number of cores used in parallel is set to the `numIslands` for convenience.
`tol`	An numerical value with the default value of `1e-05`. The tolerance level that helps evaluate whether the two iterations have the same fitness value, which aids in determining GA termination.
`seed`	An integer with the default value of `NULL`. An single integer allows function produce reproducible results.
`popInitialize`	A function or sourced function name character string. It should be designed for initializing a population. The default population initialization is random initialization with some imposed constraints. See `random_population` for example. The function returned object is a matrix, `pop`. The users can specified their own `population` function. It could also be a matrix object, which contain the user specified chromosome. By default, each column represents one individual chromosome. See `random_population` for details.
`suggestions`	A list object. Default value is `NULL`. Each element includes better or more reasonable guessed changepoints locations, which will resulting in one chromosome. If the number of provided suggestions equals to `popSize`, all the suggestions will be used as population. If the number of provided suggestions is less than `popSize`, the function from `popInitialize` will generate the remaining individual chromosomes. The number of provided suggestions cannot be greater than `popSize`. Having better `suggestions` can help GA converges faster.
`selection`	A function or sourced function name character string. This GA operator can help select `mom` and `dad` from current generation population, where `dad` is set to have better fit (smaller fitness function values). The default for selection uses the linear rank selection method. See `selection_linear_rank` for example. The function returned object is a list contain the chromosomes for `mom` and `dad`.
`crossover`	A function or sourced function name character string. This GA operator can apply crossover to the chosen parents to produce child for next generation with

	specified probability. The default for crossover uses the uniform crossover method. See `uniform_crossover` for details in the default crossover operator. The function returned object is a vector contain the chromosomes for `child`.
`mutation`	A function or sourced function name character string. This GA operator can apply mutation to the produced `child` with the specified probability `pmutation`. See `mutation` for details in the default mutation operator. The function returned object is a vector contain `child` chromosome representation.
`...`	additional arguments that will be passed to the fitness function.

Details

For any pre-specified time series model with a specified set of changepoint locations, model fit is evaluated using a fitness function $Q(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{s}, \boldsymbol{\tau}, \boldsymbol{\beta})'$ denotes the full parameter vector. Here, $\boldsymbol{s}$ denotes the set of model hyperparameters, potentially including the AR or MA orders, the degree of ARIMA differencing, or the periodic AR order for PAR models. The vector $\boldsymbol{\tau} = \{\tau_1, \dots, \tau_m\}$ specifies the changepoint locations, with the number of changepoints m inferred as part of the estimation. Each individual chromosome representation is specified as a vector,

$$C = (m, \boldsymbol{s}, \boldsymbol{\tau}, N + 1)',$$

where m represents the number of changepoints and is also equivalent to the length of vector $\boldsymbol{\tau}$ containing all the changepoint locations. $\boldsymbol{s}$ contains the integer-valued parameter for time series model structure specification, such as AR, MA, or PAR orders. If no model order selection is desired, then $\boldsymbol{s}$ is omitted and the GA detects changepoints only. The changepoint locations in $\boldsymbol{\tau}$ are encoded as interger values between 2 and N, allowing the length of $\boldsymbol{\tau}$ to vary dynamically with m. The value $N + 1$ is appended at the end of C to serve as a delimiter marking the boundary of the chromosome.

Value

Return an object class `cptgaisl-class`. See `cptgaisl-class` for a more detailed description.

Examples

```
Ts <- 1000
XMatT <- matrix(1, nrow = Ts, ncol = 1)
Xt <- ts_sim(
  Ts = Ts, beta = 0.5, XMat = XMatT, sigma = 1, phi = 0.5, theta = NULL,
  Delta = c(2, -2), CpLoc = c(250, 750), seed = 1234
)

## Multiple changepoint detection without model order selection

# without suggestions
GAISLres <- cptgaisl(ObjFunc = arima_bic, N = Ts, XMat = XMatT, Xt = Xt)
summary(GAISLres)
plot(GAISLres, data = Xt)

# with suggestions
```

```
suggestions <- list(NULL, 250, c(250, 500), c(250, 625), c(250, 500, 750))
GAISLres <- cptgaisl(ObjFunc = arima_bic, N = Ts, suggestions = suggestions, XMat = XMatT, Xt = Xt)
summary(GAISLres)
plot(GAISLres, data = Xt)


## Multiple changepoint detection with model order selection

prange <- list(ar = c(0, 3), ma = c(0, 3))

# without suggestions
GAISLres <- cptgaisl(
  ObjFunc = arima_bic_order_pq, N = Ts, prange = prange,
  option = "both", XMat = XMatT, Xt = Xt
)
summary(GAISLres)
plot(GAISLres, data = Xt)

# with suggestions
suggestions <- list(NULL, 250, c(250, 500), c(250, 625), c(250, 500, 750))
GAISLres <- cptgaisl(
  ObjFunc = arima_bic_order_pq, N = Ts, prange = prange,
  suggestions = suggestions, option = "both", XMat = XMatT, Xt = Xt
)
summary(GAISLres)
plot(GAISLres, data = Xt)
```

`cptgaisl-class` *S4 Class Definition for ‘cptgaisl‘*

Description

S4 Class for Island Model Genetic Algorithm-Based Changepoint Detection

Usage

```
## S4 method for signature 'cptgaisl'
summary(object, ...)
```

Arguments

`object`	An object of class `cptgaisl`.
`...`	Additional arguments (ignored).

Details

This class stores results and settings for the island model genetic algorithm (‘cptgaisl‘) used in multiple changepoint detection and optional model order selection.

Value

An object of class `cptgaisl`

Slots

`call` The matched call that created the object.

`N` The sample size of the time series.

`prange` A list or NULL. Ranges for each model order parameter when `option = "both"`.

`popSize` Integer. It represents the total number of individuals in each generation, which equal to the number of islands multiplied by the size of each island (i.e., `popSize = numIslands × Islandsize`).

`numIslands` Integer. The number of islands (sub-populations).

`Islandsize` Numerical value. The number of individuals in each island.

`pcrossover` Probability of crossover operation.

`pmutation` Probability of mutation operation.

`pchangepoint` Prior probability of changepoint occurrence.

`minDist` Minimum distance between adjacent changepoints.

`mmax` Maximum number of changepoints allowed.

`lmax` Maximum length of chromosome.

`maxMig` Maximum number of migrations allowed.

`maxgen` Maximum number of generations per island before migration.

`maxconv` Number of migrations with no improvement before stopping.

`option` Either "cp" or "both".

`monitoring` Logical. If TRUE, prints intermediate output.

`parallel` Logical. Whether parallel computation is used.

`nCore` Integer or NULL. Number of cores for parallelization.

`tol` Tolerance threshold for fitness improvement.

`seed` Integer or NULL. Seed for reproducibility.

`suggestions` A list or NULL. Suggested changepoint configurations.

`Island` A 3D array storing all individual chromosomes in current generation across islands. Dimensions are `lmax × Islandsize × numIslands`, representing chromosome length, individuals per island, and number of islands, respectively.

`IslandFit` A matrix of fitness values in current generation with dimensions `Islandsize × numIslands`, where each column corresponds to one island's population.

`overbestchrom` A vector. The best chromosome ever found.

`overbestfit` Numeric. The best fitness score obtained.

`bestfit` Numeric vector recording best fitness per migration.

`countMig` Integer vector tracking number of migrations.

`count` Integer vector tracking total generations.

`convg` Integer vector for convergence diagnostics.

See Also

cptgaisl, cptgaisl-class, random_population, selection_linear_rank, uniform_crossover, mutation.

`cpt_dist` *Comparing multiple changepoint configurations by pairwise distance*

Description

This function is to calculate the distance metric (Shi et al., 2022) between two changepoint configurations, $C_1 = \{\tau_1, \ldots, \tau_m\}$ and $C_2 = \{\eta_1, \ldots, \eta_k\}$, where τ's are the changepoint locations.

Usage

```
cpt_dist(tau1, tau2, N)
```

Arguments

tau1	A vector contains the changepoint locations for C_1 for comparison. A value NULL is required if there is no changepoint detected.
tau2	A vector contains the changepoint locations for C_2 for comparison. A value NULL is required if there is no changepoint detected.
N	The simulated time series sample size. Two changepoint configurations should have the same n values.

Details

The pairwise distance was proposed by Shi et al. (2022),

$$d(C_1, C_2) = |m - k| + \min(A(C_1, C_2)),$$

where m is the number of changepoints in configuration C_1 and k is the number of changepoints in configuration C_2. The term $\min(A(C_1, C_2))$ reflects the cost of matching changepoint locations between C_1 and C_2 and can be calculated using the linear assignment method. Details can be found in Shi et al. (2022). Note: if one configuration doesn't contain any changepoints (valued NULL), the distance is defined as $|m - k|$. The function can also be used to examine changepoint detection performance in simulation studies. Given the true changepoint configuration, C_{true}, used in generating the time series, the calculated distance between the estimated multiple changepoint configuration, $C_{est} = \{\eta_1, \ldots, \eta_k\}$, and C_{true} can be used to evaluate the performance of the detection algorithm.

Value

dist	The calculated distance.

References

Shi, X., Gallagher, C., Lund, R., & Killick, R. (2022). A comparison of single and multiple changepoint techniques for time series data. *Computational Statistics & Data Analysis*, 170, 107433.

Examples

```
N <- 100

# both tau1 and tau2 has detected changepoints
tau2 <- c(25, 50, 75)
tau1 <- c(20, 35, 70, 80, 90)
cpt_dist(tau1 = tau1, tau2 = tau2, N = N)

# either tau1 or tau2 has zero detected changepoints
cpt_dist(tau1 = tau1, tau2 = NULL, N = N)
cpt_dist(tau1 = NULL, tau2 = tau2, N = N)

# both tau1 and tau2 has zero detected changepoints
cpt_dist(tau1 = NULL, tau2 = NULL, N = N)
```

`mutation` *The default mutation operator in genetic algorithm*

Description

In a certain probability, the `mutation` genetic operator can be applied to generate a new `child`. By default, the new child selection can be down by the similar individual selection method in population initialization, `select_tau`.

Usage

```
mutation(child, prange = NULL, minDist, pchangepoint, lmax, mmax, N)
```

Arguments

`child`	The child chromosome resulting from the `crossover` genetic operator.
`prange`	Default is `NULL` for only changepoint detection. If `prange` is specified as a list object, which contains the range of each model order parameters for order selection (integers). The number of order parameters must be equal to the length of `prange`.
`minDist`	The required minimum distance between two adjacent changepoints.
`pchangepoint`	The probability of changepoints for every time series.
`lmax`	The user specified maximum number of changepoints, by default, as `N/2 - 1`.
`mmax`	The user specified maximum length of individual chromosome, by default, as `2+N/2 + 1`.
`N`	The sample size of the time series.

Details

A function can apply mutation to the produced child with the specified probability `pmutation` in `cptga` and `cptgaisl`. If order selection is not requested (`option = "cp"` in `cptga` and `cptgaisl`), the default `mutation` operator function uses `select_tau` to select a completely new individual with a new chromosome as the mutated child. For details, see `select_tau`. If order selection is needed (`option = "both"` in `cptga` and `cptgaisl`), we first decide whether to keep the produced child's model order with a probability of 0.5. If the child's model order is retained, the `select_tau` function is used to select a completely new individual with a new chromosome as the mutated child. If a new model order is selected from the candidate model order set, there is a 0.5 probability to either select a completely new individual with new changepoint locations or retain the original child's changepoint locations for the mutated child. Note that the current model orders in the child's chromosome are excluded from the set to avoid redundant objective function evaluation. Finally, the function returns a vector containing the modified chromosomes for mutated `child`.

Value

The resulting child chromosome representation.

`plot.cptga` *Plot Time Series with Detected Changepoints from a 'cptga' Object*

Description

This function visualizes a univariate time series along with the changepoints identified by a basic genetic algorithm, as represented by a 'cptga' object. Vertical dashed lines mark changepoint locations, and segment means are shown as horizontal dashed lines. The optimal fitness value and changepoint locations are displayed as margin text.

Usage

```
## S3 method for class 'cptga'
plot(
  x,
  data,
  main = NULL,
  XTickLab = NULL,
  XTickPos = NULL,
  XAxisLab = "Time",
  YAxisLab = "Data",
  cex.lab = 1.3,
  cex.axis = 1.3,
  cex.main = 1.3,
  lwd = 2,
  ...
)
```

Arguments

`x`	An object of class `cptgaisl`, typically returned by a basic genetic algorithm based changepoint detection procedure.
`data`	A numeric vector representing the observed univariate time series.
`main`	Optional main title for the plot.
`XTickLab`	Optional vector (e.g., numeric or date) for custom x-axis labels. Must be the same length as `data`.
`XTickPos`	Optional vector specifying which elements of `XTickLab` to show as ticks.
`XAxisLab`	Optional label for the x-axis. Default is `"Time"`.
`YAxisLab`	Optional label for the y-axis. Default is `"Data"`.
`cex.lab`	Text size for axis labels and margin text. Default is `1.3`.
`cex.axis`	Text size for axis tick labels. Default is `1.3`.
`cex.main`	Text size for the main title. Default is `1.3`.
`lwd`	Line width for vertical and horizontal dashed lines. Default is `2`.
`...`	Additional graphical parameters passed to `plot()`.

Details

If `XTickLab` is supplied and matches the length of `data`, it is used for the x-axis; otherwise, the default sequence `1:length(data)` is used.

If the genetic algorithm was run with `option = "both"`, the function skips hyperparameters in the chromosome when extracting changepoint positions.

The plot displays vertical dashed lines at changepoint locations and horizontal dashed lines for the mean of each segment. Fitness and changepoint summaries are shown above the plot.

Value

This function is called for its side effects and returns `NULL` invisibly.

See Also

`cptga`, `summary`, `plot.cptga`

`plot.cptgaisl` *Plot Time Series with Detected Changepoints from a 'cptgaisl' Object*

Description

This function visualizes a univariate time series along with the changepoints identified by a island model genetic algorithm, as represented by a 'cptgaisl' object. Vertical dashed lines mark changepoint locations, and segment means are shown as horizontal dashed lines. The optimal fitness value and changepoint locations are displayed as margin text.

Usage

```
## S3 method for class 'cptgaisl'
plot(
  x,
  data,
  main = NULL,
  XTickLab = NULL,
  XTickPos = NULL,
  XAxisLab = "Time",
  YAxisLab = "Data",
  cex.lab = 1.3,
  cex.axis = 1.3,
  cex.main = 1.3,
  lwd = 2,
  ...
)
```

Arguments

`x`	An object of class `cptgaisl`, typically returned by an island model genetic algorithm based changepoint detection procedure.
`data`	A numeric vector representing the observed univariate time series.
`main`	Optional main title for the plot.
`XTickLab`	Optional vector (e.g., numeric or date) for custom x-axis labels. Must be the same length as `data`.
`XTickPos`	Optional vector specifying which elements of `XTickLab` to show as ticks.
`XAxisLab`	Optional label for the x-axis. Default is `"Time"`.
`YAxisLab`	Optional label for the y-axis. Default is `"Data"`.
`cex.lab`	Text size for axis labels and margin text. Default is `1.3`.
`cex.axis`	Text size for axis tick labels. Default is `1.3`.
`cex.main`	Text size for the main title. Default is `1.3`.
`lwd`	Line width for vertical and horizontal dashed lines. Default is `2`.
`...`	Additional graphical parameters passed to `plot()`.

Details

If `XTickLab` is supplied and matches the length of `data`, it is used for the x-axis; otherwise, the default sequence `1:length(data)` is used.

If the genetic algorithm was run with `option = "both"`, the function skips hyperparameters in the chromosome when extracting changepoint positions.

The plot displays vertical dashed lines at changepoint locations and horizontal dashed lines for the mean of each segment. Fitness and changepoint summaries are shown above the plot.

Value

This function is called for its side effects and returns `NULL` invisibly.

See Also

summary, print.summary.cptgaisl

---

print.summary.cptga *Print Summary for a 'cptga' Object*

Description

Displays key information about the settings and results from a changepoint detection procedure using the Genetic Algorithm (GA) stored in a 'cptga' object. This includes the algorithm configuration, population settings, optimization mode, and final solution such as the number and location of changepoints and model parameters (if applicable).

Usage

```
## S3 method for class 'summary.cptga'
print(x, digits = getOption("digits"), max_display = 5, ...)
```

Arguments

`x`	An object of class `cptga`, typically produced by a GA-based changepoint detection routine.
`digits`	Number of digits to print for probabilities and fitness. Default taken from `getOption("digits")`.
`max_display`	Maximum number of suggested solutions to display if suggestions are provided.
`...`	Additional arguments (currently not used).

Details

When the GA is run in `option = "cp"` mode, only changepoint locations are shown. If `option = "both"`, the output includes the selected model hyperparameters along with changepoint locations.

The function uses plain text output to print a formatted summary to the console. If `x@suggestions` is provided, only up to `max_display` suggestions will be shown.

Value

Invisibly returns `NULL`. Called for its side effect of printing to the console.

See Also

cptga, summary, plot.cptga

`print.summary.cptgaisl` *Print Summary for a ‘cptgaisl‘ Object*

Description

Displays key information about the settings and results from a changepoint detection procedure using the Island model Genetic Algorithm (IMGA) stored in a ‘cptgaisl‘ object. This includes the algorithm configuration, population settings, optimization mode, and final solution such as the number and location of changepoints and model parameters (if applicable).

Usage

```
## S3 method for class 'summary.cptgaisl'
print(x, digits = getOption("digits"), max_display = 5, ...)
```

Arguments

`x`	An object of class `cptgaisl`, typically produced by a GA-based changepoint detection routine.
`digits`	Number of digits to print for probabilities and fitness. Default taken from `getOption("digits")`.
`max_display`	Maximum number of suggested solutions to display if suggestions are provided.
`...`	Additional arguments (currently not used).

Details

When the GA is run in `option = "cp"` mode, only changepoint locations are shown. If `option = "both"`, the output includes the selected model hyperparameters along with changepoint locations.

The function uses plain text output to print a formatted summary to the console. If `x@suggestions` is provided, only up to `max_display` suggestions will be shown.

Value

Invisibly returns `NULL`. Called for its side effect of printing to the console.

See Also

`cptgaisl`, `summary`, `plot.cptgaisl`

random_population *Random population initialization*

Description

Randomly generate the individuals' chromosomes (changepoint confirgurations) to construct the first generation population.

Usage

```
random_population(popSize, prange, N, minDist, pchangepoint, mmax, lmax)
```

Arguments

popSize	An integer represents the number of individual in each population for GA (or subpopulation for IslandGA).
prange	Default is `NULL` for only changepoint detection. If `prange` is specified as a list object, which contains the range of each model order parameters for order selection (integers). The number of order parameters must be equal to the length of `prange`.
N	The length of time series.
minDist	The minimum length between two adjacent changepoints.
pchangepoint	Same as `pchangepoint` from `cptga` or `cptgaisl`, the probability that a changepoint has occurred.
mmax	The maximum possible number of changepoints in the data set.
lmax	The maximum possible length of the chromosome representation.

Details

Each population can be stored in a matrix with `lmax` rows and `popSize` columns, where each column represents an individual chromosome in the format of

$$C = (m, \boldsymbol{s}, \boldsymbol{\tau}, N + 1)',$$

in `cptga` or `cptgaisl`. This function can randomly initialize the population matrix with some imposed constraints, such as ensuring the number of time points between two adjacent changepoints is greater than or equal to `minDist`. This prevents unrealistic scenario of two changepoints being too close to each other and helps reduce the total number of admissible solutions in the search space. Users can adjust the level of searching space reduction by setting an appropriate value for `minDist`. During population initialization, each changepoint location in $\boldsymbol{\tau}$ can be selected sequentially. With a specified probability `pchangepoint` denoting the probability of a time point being selected as a changepoint, the first changepoint τ_1 is randomly picked between $t = 1 + minDist$ and $t = N$. Then τ_2 is randomly selected from a smaller range between $t = \tau_1 + minDist$ and $t = N$. The process is continued until the last admissible changepoint $t = N - minDist$ is exceeded, and the number of changepoints m is obtained automatically. For added flexibility, users can specify their own population initialization function. The default population initialization uses `select_tau` to select the chromosome for the first generation population.

Value

A matrix that contains each individual's chromosome.

`selection_linear_rank` *The default parents selection genetic algorithm operator*

Description

The genetic algorithm require to select a pair of chromosomes, representing `dad` and `mom`, for the `crossover` operator to produce offspring (individual for next generation). The parents chromosomes are randomly selectd from the initialized population by a linear ranking method according to each individual's fittness in the input argument `popFit`. By default, the dad has better fit/smaller fitness function value/larger rank than `mom`.

Usage

```
selection_linear_rank(pop, popFit)
```

Arguments

`pop`	A matrix contains the chromosomes for all individuals. The number of rows is equal to `lmax` and the number of columns is equal to the `popSize`.
`popFit`	A vector contains the objective function value (population fit) being associated to each individual chromosome from above.

Value

A list contains the chromosomes for `dad` and `mom`.

`select_tau` *Randomly select the chromosome*

Description

Randomly select the changepoint configuration for population initialization. The selected changepoint configuration represents a changepoint chromosome. The first element of the chromosome represent the number of changepoints and the last non-zero element always equal to the length of time series + 1.

Usage

```
select_tau(N, prange, minDist, pchangepoint, mmax, lmax)
```

Arguments

`N`	The length of time series.
`prange`	A list object containing the possible range for other pre-defined model parameters, i.e. AR/MA order of ARMA models.
`minDist`	The minimum length between two adjacent changepoints.
`pchangepoint`	Same as `Pchangepoint`, the probability that a changepoint has occurred.
`mmax`	The maximum possible number of changepoints in the data set.
`lmax`	The maximum possible length of the chromosome representation.

Value

A single changepoint configuration format as above.

`ts_sim` *Time series simulation with changepoint effects*

Description

This is a function to simulate time series with changepoint effects. See details below.

Usage

```
ts_sim(
  Ts,
  beta,
  XMat,
  sigma,
  phi = NULL,
  theta = NULL,
  d = 0,
  Delta = NULL,
  CpLoc = NULL,
  seed = NULL
)
```

Arguments

`Ts`	A integer indicating simulated time series length (sample size).
`beta`	A parameter vector contains other mean function parameters without changepoint parameters.
`XMat`	The covairates for time series mean function without changepoint indicators.
`sigma`	The standard deviation for time series residuals ϵ_t.
`phi`	A vector for the autoregressive (AR) parameters for AR part.
`theta`	A vector for the moving average (MA) parameters for MA part.

`d`	A nonnegative integer for the order of differencing in the ARIMA model. Default is 0, which reduces to an ARMA model.
`Delta`	The parameter vector contains the changepoint parameters for time series mean function.
`CpLoc`	A vector contains the changepoint locations range from $1 \leq \tau \leq T_s$.
`seed`	The random seed for simulation reproducibility.

Details

The simulated time series $Z_t, t = 1, \ldots, T_s$ is generated from

$$Z_t = \mu_t + e_t.$$

The error process e_t can be:

- IID Gaussian noise when `phi = NULL`, `theta = NULL`, and `d = 0`.
- An ARMA(p,q) process when `d = 0` and at least one of `phi` or `theta` is specified.
- An ARIMA(p,d,q) process when `d > 0`.

The mean structure μ_t is specified through `XMat` and `beta`, and changepoint effects can be introduced as

$$\mu_t = X_t^\top \beta + \Delta_1 I(t \geq \tau_1) + \cdots + \Delta_m I(t \geq \tau_m),$$

where $1 \leq \tau_1 < \cdots < \tau_m \leq T_s$ are changepoint locations and $\Delta_1, \ldots, \Delta_m$ are changepoint magnitudes.

Value

The simulated time series with attributes:

- `DesignX`: The covariates including changepoint indicators.
- `mu`: The mean vector for the simulated time series.
- `theta`: The true parameter vector including changepoint effects.
- `CpEff`: The true changepoint magnitudes.
- `CpLoc`: The true changepoint locations.
- `arEff`: The AR coefficients.
- `maEff`: The MA coefficients.
- `diffEff`: The differencing order.
- `seed`: The random seed used for simulation.

Examples

```
##### M1: Time series observations are IID
Ts <- 1000
betaT <- c(0.5)
XMatT <- matrix(1, nrow = Ts, ncol = 1)
colnames(XMatT) <- "intercept"
sigmaT <- 1
DeltaT <- c(2, -2)
Cpprop <- c(1 / 4, 3 / 4)
CpLocT <- floor(Ts * Cpprop)

myts <- ts_sim(
  Ts = Ts, beta = betaT, XMat = XMatT, sigma = sigmaT,
  Delta = DeltaT, CpLoc = CpLocT, seed = 1234
)

##### M2: ARMA(2,1) model with constant mean
Ts <- 1000
betaT <- c(0.5)
XMatT <- matrix(1, nrow = Ts, ncol = 1)
colnames(XMatT) <- "intercept"
sigmaT <- 1
phiT <- c(0.5, -0.5)
thetaT <- c(0.8)
DeltaT <- c(2, -2)
CpLocT <- floor(Ts * c(1 / 4, 3 / 4))

myts <- ts_sim(
  Ts = Ts, beta = betaT, XMat = XMatT, sigma = sigmaT,
  phi = phiT, theta = thetaT, d = 0,
  Delta = DeltaT, CpLoc = CpLocT, seed = 1234
)

##### M3: ARIMA(1,1,1) model with changepoint effects
Ts <- 1000
betaT <- c(0.5)
XMatT <- matrix(1, nrow = Ts, ncol = 1)
colnames(XMatT) <- "intercept"
sigmaT <- 1
phiT <- c(0.5)
thetaT <- c(-0.3)
DeltaT <- c(2, -2)
CpLocT <- floor(Ts * c(1 / 4, 3 / 4))

myts <- ts_sim(
  Ts = Ts, beta = betaT, XMat = XMatT, sigma = sigmaT,
  phi = phiT, theta = thetaT, d = 1,
  Delta = DeltaT, CpLoc = CpLocT, seed = 1234
)
```

`uniform_crossover` *Uniform crossover to produce offsprings*

Description

In uniform crossover, typically, each bit is chosen from either parent with equal probability. Other mixing ratios are sometimes used, resulting in offspring which inherit more genetic information from one parent than the other. In a uniform crossover, we don't divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it will be included in the off-spring. If model order selection is requested, each child's model order has the equal probability (0.5) from `dad` and `mom`.

Usage

```
uniform_crossover(mom, dad, prange, minDist, lmax, N)
```

Arguments

`mom`	Among two selected individuals, `mom` represents the selected chromosome representation with lower fitness function value.
`dad`	Among two selected individuals, `dad` represents the selected chromosome representation with larger fitness function value.
`prange`	Default value is `NULL` for only changepoint detection. If `prange` is specified as a list object, which contains the range of each model order parameters for order selection (integers). The number of order parameters must be equal to the length of `prange`.
`minDist`	The required minimum distance between two adjacent changepoints.
`lmax`	The user specified maximum number of changepoints, by default, as `N/2 - 1`.
`N`	The length of time series.

Value

The child chromosome that produced from `mom` and `dad` for next generation.

Index