Package 'changepointGA'

July 22, 2025

Type Package

Title Changepoint Detection via Modified Genetic Algorithm

Version 0.1.1

Date 2025-05-22

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 MIT + file LICENSE

RoxygenNote 7.3.2

Depends R (>= 4.3.0)

Imports foreach, doParallel, Rcpp, RcppArmadillo, clue, stats

LinkingTo Rcpp, RcppArmadillo

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

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

Suggests knitr, rmarkdown

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2025-05-22 21:10:06 UTC

Contents

AMOCcrossover 2
AMOCmutation 3
AMOCpopulation 4
AMOCselection 5
ARIMA.BIC 5
ARIMA.BIC.Order 6
cptDist 7
GA 8
GA_param 10
IslandGA 11
IslandGA_param 13
mutation 14
operators 15
random_population 16
selection_linearrank 17
selectTau 18
ts.sim 18
TsPlotCheck 21
uniformcrossover 22

Index **24**

`AMOCcrossover` *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

```
AMOCcrossover(mom, dad, p.range = 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.
p.range	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.

AMOCmutation *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

```
AMOCmutation(
  child,
  p.range = NULL,
  minDist,
  Pchangepoint = NULL,
  lmax = NULL,
  mmax = NULL,
  N = NULL
)
```

Arguments

child	The child chromosome resulting from the crossover genetic operator.
p.range	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 AMOCselection operator, which is also the jump magnitude in the AMOCmutation 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.

AMOCpopulation *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

```
AMOCpopulation(popsize, p.range, N, minDist, Pchangepoint, mmax, lmax)
```

Arguments

popsize	An integer represents the number of individual in each population for GA (or subpopulation for IslandGA).
p.range	Default is NULL for only changepoint detection. If p.range 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 p.range.
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.

AMOCselection *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_linearrank 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

```
AMOCselection(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(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 — *Calculating BIC for Multiple changepoint detection with model 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(chromosome, plen = 2, 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

```
N = 1000
XMatT = matrix(1, nrow=N, ncol=1)
Xt = ts.sim(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(chromosome, plen=2, XMat=XMatT, Xt=Xt)
```

---

cptDist *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

```
cptDist(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)
cptDist(tau1=tau1, tau2=tau2, N=N)

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

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

GA *Genetic algorithm*

Description

Perform the modified 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

```
GA(
  ObjFunc,
  N,
  GA_param = .default.GA_param,
  GA_operators = .default.operators,
```

```
  p.range = NULL,
  ...
)
```

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 in GA_param. When option="both", the list p.range must be specified to give the range of model orders.
N	The sample size of the time series.
GA_param	A list contains the hyper-parameters for genetic algorithm. The default values for these hyper-parameters are included in .default.GA_param. See GA_param for the details.
GA_operators	A list includes the functions for population initialization, new individual selection, and genetic operator of crossover and mutation. See operators for the details and default functions.
p.range	Default is NULL for only changepoint detection. If p.range 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 p.range.
...	additional arguments that will be passed to the fitness function.

Value

Returns a list that has components:

overbestfit	The obtained minimum value of the objective function after the final iteration.
overbestchrom	The locations of the detected changepoints associating with the overbestfit the after the final iteration.
bestfit	The minimized fitness function values at each iteration.
bestchrom	The detected changepoints at each iteration.
count	The number of iterations undertaken by the genetic algorithm.
convg	An integer code.

- 0 indicates the algorithm successful completion.
- 1 indicates the the total number of generations exceeds the prespecified maxgen limit.

Examples

```
N = 1000
XMatT = matrix(1, nrow=N, ncol=1)
Xt = ts.sim(beta=0.5, XMat=XMatT, sigma=1, phi=0.5, theta=NULL,
            Delta=c(2, -2), CpLoc=c(250, 750), seed=1234)
TsPlotCheck(X=1:N, Xat=seq(from=1, to=N, length=10), Y=Xt, tau=c(250, 750))

GA.res = GA(ObjFunc=ARIMA.BIC, N=N, XMat=XMatT, Xt=Xt)
GA.res$overbestfit
GA.res$overbestchrom
```

GA_param *GA_param*

Description

A list object contains the hyperparameters for GA running.

Arguments

popsize	An integer represents the number of individuals in each population.
Pcrossover	The probability that the crossover operator applies on two individual chromosomes.
Pmutation	The probability that the mutation operator applies on one individual chromosome.
Pchangepoint	The probability that a changepoint has occurred. User could change this probability based on domain knowledge and the time series length.
minDist	The minimum length between two adjacent changepoints.
mmax	The maximum possible number of changepoints in the data set. For a time series of length 1000, 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.
lmax	The maximum possible length of the chromosome representation. For a time series of length 1000, 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.
maxgen	The maximum number of generation that the GA can last.
maxconv	If the overall best fitted value doesn't change after maxconv consecutive migrations, the GA algorithm stops.
option	A 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 binary interger 0 or 1, indicating whether print out middle results for each iterations of GA.

parallel	Whether use multiple threads to parallel compute the individual fittness function values.
nCore	An integer represents the number of cores used in parallel computing.
tol	The tolerance level for deciding GA to stop.
seed	An single integer allows function produce reproducible results.

Author(s)

Mo Li

Examples

```
# time series length
N = 1000

GA_param = list(
  popsize      = 40,
  Pcrossover   = 0.95,
  Pmutation    = 0.1,
  Pchangepoint = 0.06,
  minDist      = 3,
  mmax         = N/2 - 1,
  lmax         = 2 + N/2 - 1,
  maxgen       = 50000,
  maxconv      = 5000,
  option       = "cp",
  monitoring   = FALSE,
  parallel     = FALSE,
  nCore        = NULL,
  tol          = 1e-5,
  seed         = NULL
)
```

IslandGA *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

```
IslandGA(
  ObjFunc,
  N,
  IslandGA_param = .default.IslandGA_param,
```

```
  IslandGA_operators = .default.operators,
  p.range = NULL,
  ...
)
```

Arguments

ObjFunc	The fitness function to be minimized. Users can specify any R or Rcpp functions as the fitness function with setting input as potential solution to the optimization problem and returning a numerical value as the output/fitness.
N	The sample size of the time series.
IslandGA_param	A list contains the hyper-parameters for IslandGA. The default values for these hyper-parameters are included in .default.IslandGA_param. See IslandGA_param for the details.
IslandGA_operators	A list includes the functions for population initialization, new individual selection, and genetic operator of crossover and mutation. See operators for the details and default functions.
p.range	Default is NULL for only changepoint detection. If p.range 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 p.range.
...	additional arguments that will be passed to the fitness function.

Value

Returns a list that has the following components.

overbestfit	The obtained minimum value of the objective function after the final iteration.
overbestchrom	The locations of the detected changepoints associating with the overbestfit the after the final iteration.
bestfit	The minimized fitness function values at each iteration.
bestchrom	The detected changepoints at each iteration.
countMig	The number of migrations undertaken by the IslandGA.
count	The number of iterations (generations) undertaken by the island genetic algorithm model.
convg	An integer code. • 0 indicates the algorithm successful completion. • 1 indicates the the total number of generations exceeds the prespecified maxgen limit.

Examples

```
N = 1000
XMatT = matrix(1, nrow=N, ncol=1)
Xt = ts.sim(beta=0.5, XMat=XMatT, sigma=1, phi=0.5, theta=NULL,
```

```
            Delta=c(2, -2), CpLoc=c(250, 750), seed=1234)
TsPlotCheck(X=1:N, Xat=seq(from=1, to=N, length=10), Y=Xt, tau=c(250, 750))

IslandGA.res = IslandGA(ObjFunc=ARIMA.BIC, N=N, XMat=XMatT, Xt=Xt)
IslandGA.res$overbestfit
IslandGA.res$overbestchrom
```

IslandGA_param *IslandGA_param*

Description

A list object contains the hyperparameters for island-based GA running.

Arguments

subpopsize	An integer represents the number of individual in each sub-population (island).
Islandsize	The number of subpopulation (island).
Pcrossover	The probability that the crossover operator applies on two individual chromosomes.
Pmutation	The probability that the mutation operator applies on one individual chromosome.
Pchangepoint	The probability that a changepoint has occurred.
minDist	The minimum length between two adjacent changepoints.
mmax	The maximum possible number of changepoints in the data set.
lmax	The maximum possible length of the chromosome representation.
maxMig	The maximum number of migrations. After maxMig migrations, the island-based GA algorithm stops.
maxgen	The maximum number of generations that each subpopulation (island) has. The migration will apply after maxgen generations for each subpopulation(island).
maxconv	If the overall best fitted value doesn't change after maxconv consecutive migrations, the island-based GA algorithm stops.
option	A 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 binary interger 0 or 1, indicating whether print out middle results for each iterations of GA.
parallel	Whether use multiple threads to parallel compute the individual fittness function values.
nCore	An integer represents the number of cores used in parallel computing.
tol	The tolerance level for deciding GA to stop.
seed	An single integer allows function produce reproducible results.

Author(s)

Mo Li

Examples

```
# time series length
N = 1000

IslandGA_param = list(
  subpopsize   = 40,
  Islandsize   = 5,
  Pcrossover   = 0.95,
  Pmutation    = 0.15,
  Pchangepoint = 0.1,
  minDist      = 2,
  mmax         = N/2 - 1,
  lmax         = 2 + N/2 - 1,
  maxMig       = 1000,
  maxgen       = 50,
  maxconv      = 100,
  option       = "cp",
  monitoring   = FALSE,
  parallel     = FALSE, ###
  nCore        = NULL,
  tol          = 1e-5,
  seed         = NULL
)
```

`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, `selectTau`.

Usage

```
mutation(child, p.range = NULL, minDist, Pb, lmax, mmax, N)
```

Arguments

`child`	The child chromosome resulting from the `crossover` genetic operator.
`p.range`	Default is `NULL` for only changepoint detection. If `p.range` 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 `p.range`.

minDist	The required minimum distance between two adjacent changepoints.
Pb	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 GA_param and IslandGA_param. If order selection is not requested (option = "cp" in GA_param and Island_GA), the default mutation operator function uses selectTau to select a completely new individual with a new chromosome as the mutated child. For details, see selectTau. If order selection is needed (option = "both" in GA_param and Island_GA), 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 selectTau 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.

---

operators *operators*

Description

A list object contains the GA or IslandGA operators function names.

Arguments

population	It could be a function or a matrix. The function 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.
selection	A function 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_linearrank for example. The function returned object is a list contain the chromosomes for mom and dad.

crossover	A function 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 uniformcrossover for details in the default crossover operator. The function returned object is a vector contain the chromosomes for child.
mutation	A function can apply mutation to the produced child with the specified probability Pmutation in GA_param and IslandGA_param. See mutation for details in the default mutation operator.

References

Lu, Q., Lund, R., & Lee, T. C. (2010). An MDL approach to the climate segmentation problem. *Ann. Appl. Stat.* 4 (1) 299 - 319.

See Also

See Also as GA.

Examples

```
operators = list(population = "random_population",
                 selection  = "selection_linearrank",
                 crossover  = "uniformcrossover",
                 mutation   = "mutation")
```

---

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, Pb, 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.

Pb	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.

Details

The default population initialization uses selectTau to select the chromosome for the first generation population. Each column from the produced population matrix represent an chromosome of an individual. 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.

---

selection_linearrank *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_linearrank(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.

selectTau *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

```
selectTau(N, prange, minDist, Pb, 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.
Pb	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(
  beta,
  XMat,
  sigma,
  phi = NULL,
  theta = NULL,
  Delta = NULL,
  CpLoc = NULL,
  seed = NULL
)
```

Arguments

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.
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 from a class of models,

$$Z_t = \mu_t + e_t.$$

- Time series observations are IID
  μ_t is a constant and e_t's are independent and identically distributed as $N(0, \sigma)$.
- ARMA(p,q) model with constant mean
  μ_t is a constant, and e_t follows an ARMA(p,q) process.
- ARMA(p,q) model with seasonality
  μ_t is not a constant,
  $$\mu_t = ASin(\frac{2\pi t}{S}) + BSin(\frac{2\pi t}{S}),$$
  and e_t follows an ARMA(p,q) process.
- ARMA(p,q) model with seasonality and trend
  μ_t is not a constant,
  $$\mu_t = ASin(\frac{2\pi t}{S}) + BSin(\frac{2\pi t}{S}) + \alpha t,$$
  and e_t follows an ARMA(p,q) process. The changepoint effects could be introduced through the μ_t as
  $$\mu_t = \Delta_1 I_{t>\tau_1} + \ldots + \Delta_m I_{t>\tau_m},$$
  where $1 \leq \tau_1 < \ldots < \tau_m \leq T_s$ are the changepoint locations and $\Delta_1, \ldots, \Delta_m$ are the changepoint parameter that need to be estimated.

Value

The simulated time series with attributes:

Z	The simulated time series.
Attributes	• DesignX The covariates include all changepoint indicators. • mu A vector includes the mean values for simulated time series sequences. • theta The true parameter vector (including changepoint effects).

- CpEff The true changepoint magnitudes.
- CpLoc The true changepoint locations where we introduce the mean changing effects.
- arEff A vector givies the AR coefficients.
- maEff A vector givies the MA coefficients.
- seed The random seed used for this simulation.

Examples

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

myts = ts.sim(beta=betaT, XMat=XMatT, sigma=sigmaT,
              Delta=DeltaT, CpLoc=CpLocT, seed=1234)
TsPlotCheck(Y=myts, tau=CpLocT)

##### M2: ARMA(2,1) model with constant mean
Ts = 1000
betaT = c(0.5) # intercept
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)
Cp.prop = c(1/4, 3/4)
CpLocT = floor(Ts*Cp.prop)

myts = ts.sim(beta=betaT, XMat=XMatT, sigma=sigmaT,
              phi=phiT, theta=thetaT, Delta=DeltaT, CpLoc=CpLocT, seed=1234)
TsPlotCheck(Y=myts, tau=CpLocT)

##### M3: ARMA(2,1) model with seasonality
Ts = 1000
betaT = c(0.5, -0.5, 0.3) # intercept, B, D
period = 30
XMatT = cbind(rep(1, Ts), cos(2*pi*(1:Ts)/period), sin(2*pi*(1:Ts)/period))
colnames(XMatT) = c("intercept", "Bvalue", "DValue")
sigmaT = 1
phiT = c(0.5, -0.5)
thetaT = c(0.8)
DeltaT = c(2, -2)
Cp.prop = c(1/4, 3/4)
CpLocT = floor(Ts*Cp.prop)
```

```
myts = ts.sim(beta=betaT, XMat=XMatT, sigma=sigmaT,
              phi=phiT, theta=thetaT, Delta=DeltaT, CpLoc=CpLocT, seed=1234)
TsPlotCheck(Y=myts, tau=CpLocT)


##### M4: ARMA(2,1) model with seasonality and trend
# scaled trend if large number of sample size
Ts = 1000
betaT = c(0.5, -0.5, 0.3, 0.01) # intercept, B, D, alpha
period = 30
XMatT = cbind(rep(1, Ts), cos(2*pi*(1:Ts)/period), sin(2*pi*(1:Ts)/period), 1:Ts)
colnames(XMatT) = c("intercept", "Bvalue", "DValue", "trend")
sigmaT = 1
phiT = c(0.5, -0.5)
thetaT = c(0.8)
DeltaT = c(2, -2)
Cp.prop = c(1/4, 3/4)
CpLocT = floor(Ts*Cp.prop)

myts = ts.sim(beta=betaT, XMat=XMatT, sigma=sigmaT,
              phi=phiT, theta=thetaT, Delta=DeltaT, CpLoc=CpLocT, seed=1234)
TsPlotCheck(Y=myts, tau=CpLocT)
```

`TsPlotCheck` *Plot the simulated time series*

Description

This is a function to plot the simulated time series with segmentation visualization by provided changepoint locations.

Usage

```
TsPlotCheck(
  X = NULL,
  Xat = NULL,
  Y,
  tau = NULL,
  mu = NULL,
  XLAB = NULL,
  YLAB = NULL
)
```

Arguments

`X`	The time series time index, which could be specified as years, months, days, or others. The default value is NULL and the vector from 1 to the time series length will be applied.
`Xat`	The values from `X` that will be used as the X axis tick marks.

Y	The time series data.
tau	The provided changepoint locations.
mu	The provided meam values for each time t.
XLAB	A descriptive label for X axis.
YLAB	A descriptive label for Y axis.

Value

No return value, called for side effects

Examples

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

myts = ts.sim(beta=betaT, XMat=XMatT, sigma=sigmaT, Delta=DeltaT, CpLoc=CpLocT, seed=1234)
TsPlotCheck(X=1:Ts, Xat=seq(from=1, to=Ts, length=10), Y=myts, tau=CpLocT)
```

---

uniformcrossover *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

```
uniformcrossover(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

AMOCcrossover, 2
AMOCmutation, 3
AMOCpopulation, 4
AMOCselection, *3*, 5
ARIMA.BIC, 5
ARIMA.BIC.Order, 6

cptDist, 7

GA, 8, *16*
GA_param, *9*, 10, *15, 16*

IslandGA, 11
IslandGA_param, *12*, 13, *15, 16*

mutation, 14, *15, 16*

operators, *9*, *12*, 15

random_population, *15*, 16

selection_linearrank, *5*, *15*, 17
selectTau, *14, 15*, *17*, 18

ts.sim, 18
TsPlotCheck, 21

uniformcrossover, *16*, 22