Package ‘gratis’

Type Package

Title Generating Time Series with Diverse and Controllable Characteristics

Version 0.2.1

Date 2020-10-08


LazyLoad yes

Repository CRAN

URL https://github.com/ykang/gratis

BugReports https://github.com/ykang/gratis/issues/

Depends R (>= 3.4.0)

Imports tsfeatures, doRNG, polynom, mvtnorm, forecast, dplyr, stats, tibble, utils, purrr, magrittr, GA, foreach, methods, shiny, tibble

Suggests fGarch, knitr, rmarkdown, rlang, shinydashboard

NeedsCompilation no

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

VignetteBuilder knitr

Author Yanfei Kang [aut, cre] (<https://orcid.org/0000-0001-8769-6650>), Feng Li [aut] (<https://orcid.org/0000-0002-4248-9778>), Rob Hyndman [ctb] (<https://orcid.org/0000-0002-2140-5352>), Mitchell O’Hara-Wild [ctb] (<https://orcid.org/0000-0001-6729-7695>), Bocong Zhao [ctb] (<https://orcid.org/0000-0001-8434-9047>)

Maintainer Yanfei Kang <yanfeikang@buaa.edu.cn>

Date/Publication 2020-10-08 13:50:06 UTC
**R topics documented:**

- arinf .......................... 2
- fitness TS ........................ 3
- ga TS .......................... 4
- generate msts ...................... 6
- generate TS ....................... 7
- generate TS with target .......... 8
- nsdiffs 1 ........................ 9
- pi coefficients ................... 9
- rlnorm2 ......................... 10
- rmixnorm ........................ 11
- rmixnorm TS ....................... 12
- run gratis app .................. 13
- tsgeneration .................... 13

Index 14

---

### arinf

*Compute pi coefficients from ARIMA model*

**Description**

Compute pi coefficients from ARIMA model

**Usage**

`arinf(object)`

**Arguments**

- `object`  
  An object of class "Arima"

**Value**

A vector of AR coefficients

**Author(s)**

Rob J Hyndman

**Examples**

`# Not Run`
fitness_ts

*Fitness function for time series generation.*

**Description**

Fitness function for time series generation.

**Usage**

```r
fitness_ts(
  pars, 
  features, 
  seasonal, 
  n = 120, 
  freq = 12, 
  target, 
  nComp, 
  selected.features 
)
```

**Arguments**

- `pars`: Parameters
- `features`: Time series features.
- `seasonal`: Seasonal effects.
- `n`: Length of time series
- `freq`: Frequency of time series
- `target`: Target time series features
- `nComp`: No. of components used in mixture models.
- `selected.features`: Selected features.

**Value**

NA

**Examples**

```r
# Not Run
```
Description

A revised version of genetic algorithms (R package ‘GA’) to allow for time series generation.

Usage

ga_ts(type = c("binary", "real-valued", "permutation"), fitness, ..., n, min, max, nBits, population = gaControl(type)$population, selection = gaControl(type)$selection, crossover = gaControl(type)$crossover, mutation = gaControl(type)$mutation, popSize = 50, pcrossover = 0.8, pmutation = 0.1, elitism = base::max(1, round(popSize * 0.05)), updatePop = FALSE, postFitness = NULL, maxiter = 100, run = maxiter, maxFitness = Inf, names = NULL, suggestions = NULL, optim = FALSE, optimArgs = list(method = "L-BFGS-B", poptim = 0.05, pressel = 0.5, control = list(fnscale = -1, maxit = 100)), keepBest = FALSE, parallel = FALSE, monitor = if (interactive()) { if (shiny::is.RStudio()) gaMonitor else FALSE } else { FALSE }, seed = NULL)

Arguments

- **type**: the type of genetic algorithm to be run depending on the nature of decision variables.
fitness the fitness function, any allowable R function which takes as input an individual string representing a potential solution, and returns a numerical value describing its "fitness"

... additional arguments to be passed to the fitness function.

n Length of the time series to be generated.

min a vector of length equal to the decision variables providing the lower bounds of the search space in case of real-valued or permutation encoded optimizations.

max a vector of length equal to the decision variables providing the upper bounds of the search space in case of real-valued or permutation encoded optimizations.

nBits a value specifying the number of bits to be used in binary encoded optimizations.

population an R function for randomly generating an initial population.

selection an R function performing selection, i.e. a function which generates a new population of individuals from the current population probabilistically according to individual fitness.

crossover an R function performing crossover, i.e. a function which forms offsprings by combining part of the genetic information from their parents.

mutation an R function performing mutation, i.e. a function which randomly alters the values of some genes in a parent chromosome.

popSize the population size.

pcrossover the probability of crossover between pairs of chromosomes.

pmutation the probability of mutation in a parent chromosome.

elitism the number of best fitness individuals to survive at each generation.

updatePop If set at TRUE the first attribute attached to the value returned by the user-defined fitness function is used to update the population.

postFitness a user-defined function which, if provided, receives the current ga-class object as input, performs post fitness-evaluation steps, then returns an updated version of the object which is used to update the GA search.

maxiter the maximum number of iterations to run before the GA search is halted.

run the number of consecutive generations without any improvement in the best fitness value before the GA is stopped.

maxFitness the upper bound on the fitness function after that the GA search is interrupted.

names a vector of character strings providing the names of decision variables.

suggestions a matrix of solutions strings to be included in the initial population.

optim a logical defaulting to FALSE determining whether or not a local search using general-purpose optimisation algorithms should be used.

optimArgs a list controlling the local search algorithm.

keepBest a logical argument specifying if best solutions at each iteration should be saved in a slot called bestSol.

parallel An optional argument which allows to specify if the Genetic Algorithm should be run sequentially or in parallel.

monitor a logical or an R function which takes as input the current state of the ga-class object and show the evolution of the search.

seed an integer value containing the random number generator state.
generate_msts

Value

An object of class 'ga-class'.

Examples

# Not Run

generate_msts

Generate multiple seasonal time series from random parameter spaces of the mixture autoregressive (MAR) models.

Usage

generate_msts(
  seasonal.periods = c(7, 365),
  n = 800,
  nComp = NULL,
  output_format = "list"
)

Arguments

seasonal.periods
  a vector of seasonal periods of the time series to be generated.

n
  length of the generated time series.

nComp
  number of mixing components when simulating time series using MAR models.

output_format
  An optional argument which allows to choose output format between "list" and "tsibble"

Value

a time series with multiple seasonal periods.

Examples

x <- generate_msts(seasonal.periods = c(7, 365), n = 800, nComp = 2, output_format = "list")
forecast::autoplot(x)
**generate_ts**

Generate time series from random parameter spaces of the mixture autoregressive (MAR) models.

**Description**

Generate time series from random parameter spaces of the mixture autoregressive (MAR) models.

**Usage**

```r
generate_ts(n.ts = 1, freq = 1, nComp = NULL, n = 120, output_format = "list")
```

**Arguments**

- `n.ts`: number of time series to be generated.
- `freq`: seasonal period of the time series to be generated.
- `nComp`: number of mixing components when simulating time series using MAR models.
- `n`: length of the generated time series.
- `output_format`: An optional argument which allows to choose output format between "list" and "tsibble"

**Value**

A list of time series together with the SARIMA coefficients used in each mixing component and the corresponding mixing weights.

**Author(s)**

Yanfei Kang and Feng Li

**References**

Wong, CS & WK Li (2000).

**Examples**

```r
x <- generate_ts(n.ts = 2, freq = 12, nComp = 2, n = 120)
x$N1$pars
forecast::autoplot(x$N1$x)
```
generate_ts_with_target

Generating time series with controllable features.

Description
Generating time series with controllable features.

Usage

generate_ts_with_target(
  n,
  ts.length,
  freq,
  seasonal,
  features,
  selected.features,
  target,
  parallel = TRUE,
  output_format = "list"
)

Arguments
n             number of time series to be generated.
Ts.length    length of the time series to be generated.
freq         frequency of the time series to be generated.
seasonal     0 for non-seasonal data, 1 for single-seasonal data, and 2 for multiple seasonal data.
features      a vector of function names.
selected.features selected features to be controlled.
target        target feature values.
parallel      An optional argument which allows to specify if the Genetic Algorithm should be run sequentially or in parallel.
output_format An optional argument which allows to choose output format between 'list' and 'tsibble'

Value
A time-series object of class "ts" or "msts".

Author(s)
Yanfei Kang
Examples

library(tsfeatures)
x <- generate_ts_with_target(n = 1, ts.length = 60, freq = 1, seasonal = 0,
                               features = c('entropy', 'stl_features'),
                               selected.features = c('entropy', 'trend'),
                               target = c(0.6, 0.9), parallel = FALSE)
forecast::autoplot(x)

nsdiffs1

Set the number of seasonal differences for yearly data to be -1.

Description

Set the number of seasonal differences for yearly data to be -1.

Usage

nsdiffs1(x)

Arguments

x

Univariate time series or numerical vector

Value

A numerical scalar value

Examples

# Not Run

pi_coefficients

Compute pi coefficients of an AR process from SARIMA coefficients.

Description

Convert SARIMA coefficients to pi coefficients of an AR process.

Usage

pi_coefficients(
    ar = 0,
    d = 0L,
    ma = 0,
    sar = 0,
    D = 0L,
    sma = 0,
    m = 1L,
    tol = 1e-07
)
Arguments

ar AR coefficients in the SARIMA model.
d number of differences in the SARIMA model.
ma MA coefficients in the SARIMA model.
sar seasonal AR coefficients in the SARIMA model.
D number of seasonal differences in the SARIMA model.
sma seasonal MA coefficients in the SARIMA model.
m seasonal period in the SARIMA model.
tol tolerance value used. Only return up to last element greater than tolerance.

Value

A vector of AR coefficients.

Author(s)

Rob J Hyndman

Examples

# Not Run

```
```
### Author(s)
Feng Li, Central University of Finance and Economics.

### References

---

### rmixnorm
*Generate random variables from mixture normal distribution.*

### Description
Random variables from mixture of normals.

### Usage

```r
rmixnorm(n, means, sigmas, weights)
```

### Arguments

- `n`: “integer”, numbers of samples to be generated.
- `means`: "q-by-k matrix" mean value within each component, total k components.
- `sigmas`: "q-by-q-by-k" variance covariance matrix with in each component.
- `weights`: "k-length vector" weights in each component.

### Value
"matrix".

### Author(s)
Feng Li, Central University of Finance and Economics.

### References
Villani et al 2009.

### Examples

```r
n <- 1000
means <- matrix(c(-5, 0, 5), 1)
sigmas <- array(c(1, 1, 1), c(1, 1, 3))
weights <- c(0.3, 0.4, 0.3)
out <- rmixnorm(n, means, sigmas, weights)
hist(out, breaks = 100, freq = FALSE)
```
rmixnorm_ts

Simulate AR type random variables from mixture of normal

Description
This function simulates random samples from a finite mixture of Gaussian distribution where the mean from each components are AR(p) process.

Usage
rmixnorm_ts(n, means.ar.par.list, sigmas.list, weights, yinit = 0)

Arguments
- n: number of samples.
- means.ar.par.list: parameters in AR(p) within each mixing component.
- sigmas.list: variance list.
- weights: weight in each list.
- yinit: initial values.

Value
vector of length n follows a mixture distribution.

Author(s)
Feng Li, Central University of Finance and Economics.

References

Examples
```r
n = 1000
means.ar.par.list = list(c(0, 0.8), c(0, 0.6, 0.3))
require("fGarch")
sigmas.spec <- list(garchSpec(model = list(alpha = c(0.05, 0.06)), cond.dist = "norm"),
garchSpec(model = list(alpha = c(0.05, 0.05)), cond.dist = "norm"))
sigmas.list <- lapply(lapply(sigmas.spec, garchSim, extended = TRUE, n = n),
function(x) x$sigma)
weights <- c(0.8, 0.2)
y = rmixnorm_ts(n = n, means.ar.par.list = means.ar.par.list, sigmas.list = sigmas.list,
               weights = weights)
plot(y)
```
run_gratis_app

Web Application to generate time series with controllable features.

**Description**

Web Application to generate time series with controllable features.

**Usage**

```r
run_gratis_app()
app_gratis()
```

**Examples**

```r
# Not Run
```

---

tsgeneration

**Time Series Generation**

**Description**

The tsgeneration package generates time series data based on MAR models.
Index

app_gratis (run_gratis_app), 13
arinf, 2

fitness_ts, 3

ga_ts, 4
generate_msts, 6
generate_ts, 7
generate_ts_with_target, 8

nsdiffs1, 9
pi_coefficients, 9

rlnorm2, 10
rmixnorm, 11
rmixnorm_ts, 12
run_gratis_app, 13
tsgeneration, 13