Package ‘synthesis’

November 27, 2021

Title Generate Synthetic Data from Statistical Models

Version 1.2.3

Author Ze Jiang [aut, cre] (https://orcid.org/0000-0002-3472-0829)

Maintainer Ze Jiang <ze.jiang@unsw.edu.au>


Depends R (>= 3.5.0)

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/zejiang-unsw/synthesis#readme

BugReports https://github.com/zejiang-unsw/synthesis/issues

Imports stats, MASS, graphics

Suggests zoo, knitr, WASP, NPRED, rmarkdown, testthat, devtools

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-11-27 11:10:02 UTC

R topics documented:

data.gen.affine ......................................................... 2
data.gen.ar1 .............................................................. 3
data.gen.ar4 .............................................................. 4
data.gen.ar9 .............................................................. 4
data.gen.affine

Generate an affine error model.

Description
Generate an affine error model.

Usage
data.gen.affine(nobs, a = 0, b = 1, ndim = 3, mu = 0, sd = 1)

Arguments

nobs  The data length to be generated.
a  intercept
b  slope
ndim  The number of potential predictors (default is 9).
mu  mean of error term
sd  standard deviation of error term
Value
A list of 2 elements: a vector of response (x), and a matrix of potential predictors (dp) with each column containing one potential predictor.

References

Examples
# Affine error model from paper with 3 dummy variables
data.affine<-data.gen.affine(500)
plot.ts(cbind(data.affine$x, data.affine$dp))

# AR1 model from paper with 9 dummy variables
data.ar1<-data.gen.ar1(500)
plot.ts(cbind(data.ar1$x, data.ar1$dp))

# Predictor Identifier
NPRED::stepwise.PIC(data.ar1$x, data.ar1$dp)
data.gen.ar4

Generate predictor and response data from AR4 model.

Description
Generate predictor and response data from AR4 model.

Usage
data.gen.ar4(nobs, ndim = 9)

Arguments
nobs The data length to be generated.
ndim The number of potential predictors (default is 9).

Value
A list of 2 elements: a vector of response (x), and a matrix of potential predictors (dp) with each column containing one potential predictor.

Examples
# AR4 model from paper with total 9 dimensions
data.ar4<-data.gen.ar4(500)
plot.ts(cbind(data.ar4$x, data.ar4$dp))

# Predictor Identifier
NPRED::stepwise.PIC(data.ar4$x, data.ar4$dp)

data.gen.ar9

Generate predictor and response data from AR9 model.

Description
Generate predictor and response data from AR9 model.

Usage
data.gen.ar9(nobs, ndim = 9)

Arguments
nobs The data length to be generated.
ndim The number of potential predictors (default is 9).
**Value**

A list of 2 elements: a vector of response (x), and a matrix of potential predictors (dp) with each column containing one potential predictor.

**Examples**

```r
# AR9 model from paper with total 9 dimensions
data.ar9<-data.gen.ar9(500)
plot.ts(cbind(data.ar9$x, data.ar9$dp))

# Predictor Identifier
NPRED::stepwise.PIC(data.ar9$x, data.ar9$dp)
```

---

**data.gen.blobs**  
*Gaussian Blobs*

**Description**

Gaussian Blobs

**Usage**

```r
data.gen.blobs(
  nobs = 100,  
  features = 2,  
  centers = 3,  
  sd = 1,  
  bbox = c(-10, 10),  
  do.plot = TRUE
)
```

**Arguments**

- `nobs`: The data length to be generated.
- `features`: Features of dataset.
- `centers`: Either the number of centers, or a matrix of the chosen centers.
- `sd`: The level of Gaussian noise, default 1.
- `bbox`: The bounding box of the dataset.
- `do.plot`: Logical value. If TRUE (default value), a plot of the generated Blobs is shown.
Details

This function generates a matrix of features creating multiclass datasets by allocating each class one or more normally-distributed clusters of points. It can control both centers and standard deviations of each cluster. For example, we want to generate a dataset of weight and height (two features) of 500 people (data length), including three groups, baby, children, and adult. Centers are the average weight and height for each group, assuming both weight and height are normally distributed (i.e. follow Gaussian distribution). The standard deviation (sd) is the sd of the Gaussian distribution while the bounding box (bbox) is the range for each generated cluster center when only the number of centers is given.

Value

A list of two variables, x and classes.

References


Examples

```
Blobs=data.gen.blobs(nobs=1000, features=2, centers=3, sd=1, bbox=c(-10,10), do.plot=TRUE)
```

---

**data.gen.bm**

Generate a time series of Brownian motion.

Description

This function generates a time series of one dimension Brownian motion.

Usage

```
data.gen.bm(
  x0 = 0,
  w0 = 0,
  time = seq(0, by = 0.01, length.out = 101),
  do.plot = TRUE
)
```

Arguments

- **x0**: the start value of x, with the default value 0
- **w0**: the start value of w, with the default value 0
- **time**: the temporal interval at which the system will be generated. Default seq(0,by=0.01,len=101).
- **do.plot**: a logical value. If TRUE (default value), a plot of the generated system is shown.
data.gen.BUWO

References


Examples

```r
set.seed(123)
x <- data.gen.bm()
```

---

**data.gen.BUWO**

*Generate build-up and wash-off model for water quality modeling*

Description

Generate build-up and wash-off model for water quality modeling

Usage

```r
data.gen.BUWO(nobs, k = 0.5, a = 1, m0 = 10, q = 0)
```

Arguments

- `nobs`: The data length to be generated.
- `k`: build-up coefficient (kg*t-1)
- `a`: wash-off rate constant (m-3)
- `m0`: threshold at which additional mass does not accumulate on the surface (kg)
- `q`: runoff (m³*t-1)

Value

A list of 2 elements: a vector of build-up mass (x), and a vector of wash-off mass (y) per unit time.

References


Examples

```r
# Build up model
set.seed(101)
sample = 500
# create a gamma shape storm event
q <- seq(0, 20, length.out=sample)
p <- pgamma(q, shape=9, rate = 2, lower.tail = TRUE)
p <- c(p[1], p[2:sample] - p[1:(sample - 1)])
data.tss <- data.gen.BUWO(sample, k = 0.5, a = 5, m0 = 10, q = p)
plot.ts(cbind(p, data.tss$x, data.tss$y), ylab=c("Q", "Bulid-up", "Wash-off"))
```

---

data.gen.circles

**Circles**

**Description**

Circles

**Usage**

```r
data.gen.circles(
  n,
  r_vec = c(1, 2),
  start = runif(1, -1, 1),
  s,
  do.plot = TRUE
)
```

**Arguments**

- `n`: The data length to be generated.
- `r_vec`: The radius of circles.
- `start`: The center of circles.
- `s`: The level of Gaussian noise, default 0.
- `do.plot`: Logical value. If TRUE (default value), a plot of the generated Circles is shown.

**Value**

A list of two variables, x and classes.

**Examples**

```r
Circles = data.gen.circles(n = 1000, r_vec = c(1, 2), start = runif(1, -1, 1), s = 0.1, do.plot = TRUE)
```
data.gen.Duffing

Duffing map

Description

Generates a 2-dimensional time series using the Duffing map.

Usage

```r
data.gen.Duffing(
  nobs = 5000,
  a = 2.75,
  b = 0.2,
  start = runif(n = 2, min = -0.5, max = 0.5),
  s,
  do.plot = TRUE
)
```

Arguments

- **nobs**: Length of the generated time series. Default: 5000 samples.
- **a**: The \( a \) parameter. Default: 2.75.
- **b**: The \( b \) parameter. Default: 0.2.
- **start**: A 2-dimensional vector indicating the starting values for the x and y Duffing coordinates. Default: If the starting point is not specified, it is generated randomly.
- **s**: The level of noise, default 0.
- **do.plot**: Logical value. If TRUE (default value), a plot of the generated Duffing system is shown.

Details

The Duffing map is defined as follows:

\[
x_n = y_{n-1} \\
y_n = -b \cdot x_{n-1} + a \cdot y_{n-1} - y_{n-1}^3
\]

The default selection for both \( a \) and \( b \) parameters (\( a=1.4 \) and \( b=0.3 \)) is known to produce a deterministic chaotic time series.

Value

A list with two vectors named \( x \) and \( y \) containing the x-components and the y-components of the Duffing map, respectively.

Note

Some initial values may lead to an unstable system that will tend to infinity.
References


Examples

Duffing.map=data.gen.Duffing(nobs = 1000, do.plot=TRUE)

---

data.gen.fbm

Generate a time series of fractional Brownian motion.

Description

This function generates a a time series of one dimension fractional Brownian motion.

Usage

data.gen.fbm(
  hurst = 0.95,
  time = seq(0, by = 0.01, length.out = 1000),
  do.plot = TRUE
)

Arguments

hurst the hurst index, with the default value 0.95, ranging from [0,1].
time the temporal interval at which the system will be generated. Default seq(0,by=0.01,len=1000).
do.plot a logical value. If TRUE (default value), a plot of the generated system is shown.

References


Examples

set.seed(123)
x <- data.gen.fbm()
data.gen.fm1

Friedman with independent uniform variates

Description
Friedman with independent uniform variates

Usage
data.gen.fm1(nobs, ndim = 9, noise = 1)

Arguments
- nobs: The data length to be generated.
- ndim: The number of potential predictors (default is 9).
- noise: The noise level in the time series.

Value
A list of 3 elements: a vector of response (x), a matrix of potential predictors (dp) with each column containing one potential predictor, and a vector of true predictor numbers.

Examples
###synthetic example - Friedman
#Friedman with independent uniform variates
data.fm1 <- data.gen.fm1(nobs=1000, ndim = 9, noise = 0)

#Friedman with correlated uniform variates
data.fm2 <- data.gen.fm2(nobs=1000, ndim = 9, r = 0.6, noise = 0)

plot.ts(cbind(data.fm1$x, data.fm2$x), col=c('red','blue'), main=NA, xlab=NA, ylab=c('Friedman with \n independent uniform variates', 'Friedman with \n correlated uniform variates'))

data.gen.fm2

Friedman with correlated uniform variates

Description
Friedman with correlated uniform variates

Usage
data.gen.fm2(nobs, ndim = 9, r = 0.6, noise = 0)
data.gen.gbm

Generate a time series of geometric Brownian motion.

Description

This function generates a time series of geometric Brownian motion.

Usage

```r
data.gen.gbm(
  x0 = 10,
  w0 = 0,
  mu = 1,
  sigma = 0.5,
  time = seq(0, by = 0.01, length.out = 101),
  do.plot = TRUE
)
```
data.gen.Henon

Arguments

- `x0` the start value of x, with the default value 10
- `w0` the start value of w, with the default value 0
- `mu` the interest/drifting rate, with the default value 1.
- `sigma` the diffusion coefficient, with the default value 0.5.
- `time` the temporal interval at which the system will be generated. Default `seq(0,by=0.01,len=101)`.
- `do.plot` a logical value. If TRUE (default value), a plot of the generated system is shown.

References


Examples

```r
set.seed(123)
x <- data.gen.gbm()
```

---

data.gen.Henon  
*Henon map*

Description

Generates a 2-dimensional time series using the Henon map.

Usage

```r
data.gen.Henon(
  nobs = 5000,
  a = 1.4,
  b = 0.3,
  start = runif(n = 2, min = -0.5, max = 0.5),
  s,
  do.plot = TRUE
)
```

Arguments

- `nobs` Length of the generated time series. Default: 5000 samples.
- `a` The `a` parameter. Default: 1.4.
- `b` The `b` parameter. Default: 0.3.
- `start` A 2-dimensional vector indicating the starting values for the x and y Henon coordinates. Default: If the starting point is not specified, it is generated randomly.
- `s` The level of noise, default 0.
- `do.plot` Logical value. If TRUE (default value), a plot of the generated Henon system is shown.
The Henon map is defined as follows:

\[ x_n = 1 - a \cdot x_{n-1}^2 + y_{n-1} \]
\[ y_n = b \cdot x_{n-1} \]

The default selection for both \(a\) and \(b\) parameters (\(a=1.4\) and \(b=0.3\)) is known to produce a deterministic chaotic time series.

**Value**

A list with two vectors named \(x\) and \(y\) containing the \(x\)-components and the \(y\)-components of the Henon map, respectively.

**Note**

Some initial values may lead to an unstable system that will tend to infinity.

**References**


**Examples**

Henon.map=data.gen.Henon(nobs = 1000, do.plot=TRUE)

---

**data.gen.HL**

*Generate predictor and response data: Hysteresis Loop*

**Description**

Generate predictor and response data: Hysteresis Loop

**Usage**

```r
data.gen.HL(
  nobs = 512,
  a = 0.8,
  b = 0.6,
  c = 0.2,
  m = 3,
  n = 5,
  fp = 25,
  fd,
  sd.x = 0.1,
  sd.y = 0.1
)
```
Arguments

- **nobs** The data length to be generated.
- **a** The $a$ parameter. Default: 0.8.
- **b** The $b$ parameter. Default: 0.6.
- **c** The $c$ parameter. Default: 0.2.
- **m** Positive integer for the split line parameter. If $m=1$, split line is linear; If $m$ is even, split line has a u shape; If $m$ is odd and higher than 1, split line has a chair or classical shape.
- **n** Positive odd integer for the bulging parameter, indicates degree of outward curving ($1=\text{highest level of bulging}$).
- **fp** The frequency in the generated response. $fp = 25$ used in the WRR paper.
- **fd** A vector of frequencies for potential predictors. $fd = c(3,5,10,15,25,30,55,70,95)$ used in the WRR paper.
- **sd.x** The noise level in the predictor.
- **sd.y** The noise level in the response.

Details

The Hysteresis is a common nonlinear phenomenon in natural systems and it can be numerical simulated by the following formulas:

\[
x_t = a \ast \cos(2\pi \ast f \ast t)
\]
\[
y_t = b \ast \cos(2\pi \ast f \ast t)^m - c \ast \sin(2\pi \ast f \ast t)^n
\]

The default selection for the system parameters ($a = 0.8$, $b = 0.6$, $c = 0.2$, $m = 3$, $n = 5$) is known to generate a classical hysteresis loop.

Value

A list of 3 elements: a vector of response ($x$), a matrix of potential predictors ($dp$) with each column containing one potential predictor, and a vector of true predictor numbers.

References


Examples

```r
###synthetic example - Hysteresis loop
#frequency, sampled from a given range
fd <- c(3,5,10,15,25,30,55,70,95)
data.HL <- data.gen.HL(m=3,n=5,nobs=512,fp=25,fd=fd)
plot.ts(cbind(data.HL$x,data.HL$dp))
```
data.gen.LGSS

Linear Gaussian state-space model

Description

Generates data from a specific linear Gaussian state space model of the form $x_t = \phi x_{t-1} + \sigma_v v_t$ and $y_t = x_t + \sigma_e e_t$, where $v_t$ and $e_t$ denote independent standard Gaussian random variables, i.e. $N(0,1)$.

Usage

```r
data.gen.LGSS(
  theta,
  nobs,
  start = runif(n = 1, min = -1, max = 1),
  do.plot = TRUE
)
```

Arguments

- `theta` The parameters $\theta = \{\phi, \sigma_v, \sigma_e\}$ of the LGSS model.
- `nobs` The data length to be generated.
- `start` A numeric value indicating the starting value for the time series. If the starting point is not specified, it is generated randomly.
- `do.plot` Logical value. If TRUE (default value), a plot of the generated LGSS system is shown.

Value

A list of two variables, state and response.

References


Examples

```r
data.LGSS <- data.gen.LGSS(theta=c(0.75,1.00,0.10), nobs=500, start=0)
```
**Description**

Generates a time series using the logistic map.

**Usage**

```r
data.gen.Logistic(
    nobs = 5000,
    r = 4,
    start = runif(n = 1, min = 0, max = 1),
    s,
    do.plot = TRUE
)
```

**Arguments**

- `nobs` Length of the generated time series. Default: 5000 samples.
- `r` The \( r \) parameter. Default: 4
- `start` A numeric value indicating the starting value for the time series. If the starting point is not specified, it is generated randomly.
- `s` The level of noise, default 0.
- `do.plot` Logical value. If TRUE (default value), a plot of the generated Logistic system is shown.

**Details**

The logistic map is defined as follows:

\[
x_n = r \cdot x_{n-1} \cdot (1 - x_{n-1})
\]

**Value**

A vector of time series.

**References**


**Examples**

```r
Logistic.map=data.gen.Logistic(nobs = 1000, do.plot=TRUE)
```
data.gen.Lorenz  Lorentz system

Description
Generates a 3-dimensional time series using the Lorenz equations.

Usage
```r
data.gen.Lorenz(
  sigma = 10,
  beta = 8/3,
  rho = 28,
  start = c(-13, -14, 47),
  time = seq(0, 50, length.out = 1000),
  s
)
```

Arguments
- **sigma**: The σ parameter. Default: 10.
- **beta**: The β parameter. Default: 8/3.
- **rho**: The ρ parameter. Default: 28.
- **start**: A 3-dimensional numeric vector indicating the starting point for the time series. Default: c(-13, -14, 47).
- **time**: The temporal interval at which the system will be generated. Default: time=seq(0,50,by=0.01).
- **s**: The level of noise, default 0.

Details
The Lorenz system is a system of ordinary differential equations defined as:

\[
\begin{align*}
\dot{x} &= \sigma(y - x) \\
\dot{y} &= \rho x - y - xz \\
\dot{z} &= -\beta z + xy
\end{align*}
\]

The default selection for the system parameters \((\sigma = 10, \rho = 28, \beta = 8/3)\) is known to produce a deterministic chaotic time series.

Value
A list with four vectors named `time`, `x`, `y` and `z` containing the time, the x-components, the y-components and the z-components of the Lorenz system, respectively.
Note
Some initial values may lead to an unstable system that will tend to infinity.

References

Examples
###Synthetic example - Lorenz
ts.l <- data.gen.Lorenz(sigma = 10, beta = 8/3, rho = 28, start = c(-13, -14, 47),
time = seq(0, by=0.05, length.out = 2000))

t.ts.plot(cbind(ts.l$x,ts.l$y,ts.l$z), col=c('black','red','blue'))

Description
Nonlinear system with independent/correlate covariates

Usage
data.gen.nl1(nobs, ndim = 15, r = 0.6, noise = 1)

Arguments
  nobs    The data length to be generated.
  ndim    The number of potential predictors (default is 9).
  r       Target Spearman correlation among covariates.
  noise   The noise level in the time series.

Value
A list of 3 elements: a vector of response (x), a matrix of potential predictors (dp) with each column containing one potential predictor, and a vector of true predictor numbers.

Examples
###synthetic example - Friedman
#Friedman with independent uniform variates
data.nl1 <- data.gen.nl1(nobs=1000)

#Friedman with correlated uniform variates
data.nl2 <- data.gen.nl2(nobs=1000)
Nonlinear system with Exogenous covariates

Description

Nonlinear system with Exogenous covariates

Usage

data.gen.nl2(nobs, ndim = 7, noise = 1)

Arguments

- **nobs**: The data length to be generated.
- **ndim**: The number of potential predictors (default is 9).
- **noise**: The noise level in the time series.

Value

A list of 3 elements: a vector of response (x), a matrix of potential predictors (dp) with each column containing one potential predictor, and a vector of true predictor numbers.

References


Examples

```r
### synthetic example - Friedman
#Friedman with independent uniform variates
data.nl1 <- data.gen.nl1(nobs=1000)

#Friedman with correlated uniform variates
data.nl2 <- data.gen.nl2(nobs=1000)

plot.ts(cbind(data.nl1$x, data.nl2$x), col=c('red', 'blue'), main=NA, xlab=NA, ylab=c('Nonlinear system with \n independent uniform variates', 'Nonlinear system with \n correlated uniform variates'))
```
**data.gen.norm**  
*Generate correlated normal variates*

**Description**  
Generate correlated normal variates

**Usage**  
```r  
data.gen.norm(n, mu = rep(0, 2), sd = rep(1, 2), r = 0.6, sigma)  
```

**Arguments**  
- `n`: The data length to be generated.
- `mu`: A vector giving the means of the variables.
- `sd`: A vector giving the standard deviation of the variables.
- `r`: The target Pearson correlation, default is 0.6.
- `sigma`: A positive-definite symmetric matrix specifying the covariance matrix of the variables.

**Value**  
A matrix of correlated normal variates

---

**data.gen.Rossler**  
*Rössler system*

**Description**  
Generates a 3-dimensional time series using the Rossler equations.

**Usage**  
```r  
data.gen.Rossler(  
a = 0.2,  
b = 0.2,  
w = 5.7,  
start = c(-2, -10, 0.2),  
time = seq(0, by = 0.05, length.out = 1000),  
s)  
```
Arguments

- **a** The \(a\) parameter. Default: 0.2.
- **b** The \(b\) parameter. Default: 0.2.
- **w** The \(w\) parameter. Default: 5.7.
- **start** A 3-dimensional numeric vector indicating the starting point for the time series. Default: \((-2, -10, 0.2)\).
- **time** The temporal interval at which the system will be generated. Default: \(\text{time}=\text{seq}(0,50,\text{by}=0.01)\) or \(\text{time} = \text{seq}(0,\text{by}=0.01,\text{length.out}=1000)\).
- **s** The level of noise, default 0.

Details

The Rössler system is a system of ordinary differential equations defined as:

\[
\begin{align*}
\dot{x} &= -(y + z) \\
\dot{y} &= x + a \cdot y \\
\dot{z} &= b + z \cdot (x - w)
\end{align*}
\]

The default selection for the system parameters \((a = 0.2, b = 0.2, w = 5.7)\) is known to produce a deterministic chaotic time series. However, the values \(a = 0.1, b = 0.1,\) and \(c = 14\) are more commonly used. These Rössler equations are simpler than those Lorenz used since only one nonlinear term appears (the product \(xz\) in the third equation).

Here, \(a = b = 0.1\) and \(c\) changes. The bifurcation diagram reveals that low values of \(c\) are periodic, but quickly become chaotic as \(c\) increases. This pattern repeats itself as \(c\) increases — there are sections of periodicity interspersed with periods of chaos, and the trend is towards higher-period orbits as \(c\) increases. For example, the period one orbit only appears for values of \(c\) around 4 and is never found again in the bifurcation diagram. The same phenomenon is seen with period three; until \(c = 12\), period three orbits can be found, but thereafter, they do not appear.

Value

A list with four vectors named **time**, **x**, **y** and **z** containing the time, the x-components, the y-components and the z-components of the Rössler system, respectively.

Note

Some initial values may lead to an unstable system that will tend to infinity.

References


Examples

### synthetic example - Rössler

ts.r <- data.gen.Rossler(a = 0.1, b = 0.1, w = 8.7, start = c(-2, -10, 0.2),
                        time = seq(0, by=0.05, length.out = 10000))

oldpar <- par(no.readonly = TRUE)
par(mfrow=c(1,1), ps=12, cex.lab=1.5)
plot.ts(cbind(ts.r$x, ts.r$y, ts.r$z), col=c('black', 'red', 'blue'))

par(mfrow=c(1,2), ps=12, cex.lab=1.5)
plot(ts.r$x, ts.r$y, xlab='x', ylab='y', type='l')
plot(ts.r$x, ts.r$z, xlab='x', ylab='z', type='l')
par(oldpar)

# Application to testing variance transformation method in:
data <- list(x = ts.r$z, dp = cbind(ts.r$x, ts.r$y))
dwt <- WASP::dwt.vt(data, wf="d4", J=7, method="dwt", pad="zero", boundary="periodic")

par(mfrow = c(ncol(dwt$dp), 1), mar = c(0, 2.5, 2, 1),
     oma = c(2, 1, 0, 0), # move plot to the right and up
     mgp = c(1.5, 0.5, 0), # move axis labels closer to axis
     pty = "m", bg = "transparent",
     ps = 12)

# plot(dwt$x, type="l", xlab=NA, ylab="SPI12", col="red")
# plot(dwt$x, type="l", xlab=NA, ylab="Rain", col="red")
for (i in 1:ncol(dwt$dp)) {
  ts.plot(cbind(dwt$dp[, i], dwt$dp.n[, i]),
          xlab = NA, ylab = NA,
          col = c("black", "blue"), lwd = c(1, 2))
}

data.gen.rw

Generate Random walk time series.

Description

Generate Random walk time series.

Usage

data.gen.rw(nobs, drift = 0.2, sd = 1)
data.gen.spirals

Arguments

- nobs: the data length to be generated
- drift: drift
- sd: the white noise in the data

Value

A list of 2 elements: random walk and random walk with drift

References


Examples

```r
set.seed(154)
data.rw <- data.gen.rw(200)
plot.ts(data.rw$xd, ylim=c(-5,55), main=
/quotesingle.Var/quotesingle.Var
/quotesingle.Var
/quotesingle.Var
/quotesingle.Var/quotesingle.Var

lines(data.rw$x, col=4); abline(h=0, col=4, lty=2); abline(a=0, b=.2, lty=2)
```

data.gen.spirals  Spirals

Description

Spirals

Usage

```r
data.gen.spirals(n, cycles = 1, s = 0, do.plot = TRUE)
```

Arguments

- n: The data length to be generated.
- cycles: The number of cycles of spirals.
- s: The level of Gaussian noise, default 0.
- do.plot: Logical value. If TRUE (default value), a plot of the generated Spirals is shown.

Value

A list of two variables, x and classes.

References

Examples
Spirals=data.gen.spirals(n = 2000, cycles=2, s=0.01, do.plot=TRUE)

data.gen.SW
Generate predictor and response data: Sinusoidal model

Description
Generate predictor and response data: Sinusoidal model

Usage
data.gen.SW(nobs = 500, freq = 50, A = 2, phi = pi, mu = 0, sd = 1)

Arguments
nobs The data length to be generated.
freq The frequencies in the generated response. Default freq=50.
A The amplitude of the sinusoidal series
phi The phase of the sinusoidal series
mu The mean of Gaussian noise in the variable.
sd The standard deviation of Gaussian noise in the variable.

Value
A list of time and x.

References

Examples
### Sinusoidal model
delta <- 1/12 # sampling rate, assuming monthly
period.max<- 2^5

N = 6*period.max/delta
scales<- 2^(0:5)[c(2,6)] #pick two scales
scales

### scale, period, and frequency
# freq=1/T; T=s/delta so freq = delta/s

tmp <- NULL
for(s in scales){

tmp <- cbind(tmp, data.gen.SW(nobs=N, freq = delta/s, A = 1, phi = 0, mu=0, sd = 0)$x)
}  
x <- rowSums(data.frame(tmp))
plot.ts(cbind(tmp,x), type = 'l', main=NA)

---

**data.gen.tar**

Generate a two-regime threshold autoregressive (TAR) process.

**Description**

Generate a two-regime threshold autoregressive (TAR) process.

**Usage**

```r
data.gen.tar(
  nobs,
  ndim = 9,
  phi1 = c(0.6, -0.1),
  phi2 = c(-1.1, 0),
  theta = 0,
  d = 2,
  p = 2,
  noise = 0.1
)
```

**Arguments**

- **nobs**: the data length to be generated
- **ndim**: The number of potential predictors (default is 9)
- **phi1**: the coefficient vector of the lower-regime model
- **phi2**: the coefficient vector of the upper-regime model
- **theta**: threshold
- **d**: delay
- **p**: maximum autoregressive order
- **noise**: the white noise in the data

**Details**

The two-regime Threshold Autoregressive (TAR) model is given by the following formula:

\[
Y_t = \phi_{1,0} + \phi_{1,1}Y_{t-1} + \ldots + \phi_{1,p}Y_{t-p} + \sigma_1 e_t, \text{ if } Y_{t-d} \leq r
\]

\[
Y_t = \phi_{2,0} + \phi_{2,1}Y_{t-1} + \ldots + \phi_{2,p}Y_{t-p} + \sigma_2 e_t, \text{ if } Y_{t-d} > r.
\]

where \( r \) is the threshold and \( d \) the delay.
Value

A list of 2 elements: a vector of response (x), and a matrix of potential predictors (dp) with each column containing one potential predictor.

References


Examples

# TAR2 model from paper with total 9 dimensions
data.tar<-data.gen.tar(500)
plot.ts(cbind(data.tar$x,data.tar$dp))
**data.gen.tar2**

Generate predictor and response data from TAR2 model.

**Description**

Generate predictor and response data from TAR2 model.

**Usage**

```r
data.gen.tar2(nobs, ndim = 9, noise = 0.1)
```

**Arguments**

- `nobs`: The data length to be generated.
- `ndim`: The number of potential predictors (default is 9).
- `noise`: The white noise in the data

**Value**

A list of 2 elements: a vector of response (x), and a matrix of potential predictors (dp) with each column containing one potential predictor.

**References**


**Examples**

```r
# TAR2 model from paper with total 9 dimensions
data.tar2<-data.gen.tar2(500)
plot.ts(cbind(data.tar2$x, data.tar2$dp))
```

---

**data.gen.unif**

Generate correlated uniform variates

**Description**

Generate correlated uniform variates

**Usage**

```r
data.gen.unif(n, ndim = 9, r = 0.6, sigma, method = c("pearson", "spearman"))
```
Arguments

- **n**: The data length to be generated.
- **ndim**: The number of potential predictors (default is 9).
- **r**: The target correlation, default is 0.6.
- **sigma**: A symmetric matrix of Pearson correlation, should be same as ndim.
- **method**: The target correlation type, including Pearson and Spearman correlation.

Value

A matrix of correlated uniform variates

References

Index

data.gen.affine, 2
data.gen.ar1, 3
data.gen.ar4, 4
data.gen.ar9, 4
data.gen.blobs, 5
data.gen.bm, 6
data.gen.BUWO, 7
data.gen.circles, 8
data.gen.Duffing, 9
data.gen.fbm, 10
data.gen.fm1, 11
data.gen.fm2, 11
data.gen.gbm, 12
data.gen.Henon, 13
data.gen.HL, 14
data.gen.LGSS, 16
data.gen.Logistic, 17
data.gen.Lorenz, 18
data.gen.nl1, 19
data.gen.nl2, 20
data.gen.norm, 21
data.gen.Rossler, 21
data.gen.rw, 23
data.gen.spirals, 24
data.gen.SW, 25
data.gen.tar, 26
data.gen.tar1, 27
data.gen.tar2, 28
data.gen.unif, 28