Package 'RGENERATE'

October 12, 2022

Maintainer Emanuele Cordano <emanuele.cordano@gmail.com>

License GPL (>= 2)

Title Tools to Generate Vector Time Series

Type Package

Author Emanuele Cordano

Description A method 'generate()' is implemented in this package for the random generation of vector time series according to models obtained by 'RMAWGEN', 'vars' or other packages. This package was created to generalize the algorithms of the 'RMAWGEN' package for the analysis and generation of any environmental vector time series.

Repository CRAN

URL https://github.com/ecor/RGENERATE

Date 2022-01-13

Version 1.3.7

Depends R (>= 3.5.0),RMAWGEN,magrittr

Suggests knitr,rmarkdown,testthat

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Date/Publication 2022-01-14 23:22:41 UTC

R topics documented:

gapFilling *gapFilling*

Description

It fills in a gab of a data frame by using generate method

Usage

```
gapFilling(x = NULL, ...)

## Default S3 method:
gapFilling(x, objectForGeneration = NULL, ...)

## S3 method for class 'data.frame'
gapFilling(
  x,
  objectForGeneration = NULL,
  max.filling = 2,
  nofill.code = -9999,
  ...
)
```

Arguments

x	object with gaps to fill
...	further argument for generate method
objectForGeneration	object used for generate method
max.filling	integer values: gap are filled if the previous max.filling values are not NA or nofill.code
nofill.code	Alternative value to NA which indicates the gaps which are not filled

Examples

```
set.seed(122)
NSTEP <- 1000
x <- rnorm(NSTEP)
y <- x+rnorm(NSTEP)
z <- c(rnorm(1),y[-1]+rnorm(NSTEP-1))
df <- data.frame(x=x,y=y,z=z)
var <- VAR(df,type="none")

dfobs <- df
dfobs[20:30,2] <- NA
n <- nrow(df)
```

```
gp <- gapFilling(x=dfobs,objectForGeneration=var,max.filling=2)
```

---

generate *generate*

Description

It generates a multivarite random series according to the model x

Usage

```
generate(x = NULL, ...)

## Default S3 method:
generate(
  x,
  FUN = rnorm,
  n = 100,
  K = 3,
  names = NULL,
  cov = NULL,
  gap.filling = NULL,
  ...
)

## S3 method for class 'varest'
generate(
  x,
  FUN = rnorm,
  n = 100,
  names = NULL,
  noise = NULL,
  exogen = NULL,
  xprev = NULL,
  gap.filling = NULL,
  ...
)

## S3 method for class 'varest2'
generate(
  x,
  FUN = rnorm,
```

```
  n = 100,
  names = NULL,
  noise = NULL,
  exogen = NULL,
  xprev = NULL,
  gap.filling = NULL,
  ...
)

## S3 method for class 'GPCAvarest2'
generate(
  x,
  FUN = rnorm,
  n = 100,
  names = NULL,
  noise = NULL,
  exogen = NULL,
  xprev = NULL,
  extremes = TRUE,
  type = 3,
  gap.filling = NULL,
  GPCA.row.gap.filling.option = TRUE,
  ...
)

## S3 method for class 'matrix'
generate(
  x,
  FUN = rnorm,
  n = 100,
  noise = NULL,
  xprev = NULL,
  names = NULL,
  gap.filling = NULL,
  type = c("autoregression", "covariance"),
  ...
)

## S3 method for class 'list'
generate(x, factor.series = names(x), n = NA, ...)

## S3 method for class 'MonthlyList'
generate(x, origin, n, ...)
```

Arguments

x	null object or the model used for random generation , e.g. a VAR model as a varest-class or varest2-class object. Default is NULL.

...	further arguments for FUN
FUN	random function of the probability distribution used for noise random generation. Default is rnorm. See https://CRAN.R-project.org/view=Distributions
n	number of generations requested
K	number of the variables to be generated simultaneously, i.e. the K parameters of a VAR. It is automatically detected by x, names or cov, if one of these is not NULL.
names	null object or string vectors or names of the variables to be generated simultaneously. Default is NULL.
cov	null object or covariance matrix of the random variables to be generated simultaneously. Default is NULL, not used in case this information can be detected from x.
gap.filling	data frame with time series with gabs (NA values) to be filled. Default is NULL and not considered, otherwise the method returns this data frame with NA row replaced with generated (e.g auto-regressed) values.
noise	null object or a generic external noise for x model residuals, e.g. standard white noise, for random generation with the model x. Default is NULL. If NULL the noise is automatically calculated.
exogen	null object or amatrix or data frame with exogeneous variables (predictors) id requested by x. Default is NULL
xprev	null object or initial condition of the multivariate random process to be generated. Default is NULL.
extremes	see inv_GPCA
type	character string used in some method implementations. See inv_GPCA. In the matrix implementation, default is "autoregression", i.e. the matrix is used as a vector auto-regression coefficient, if it is "covariance" the method genereted a sample with covariance matrix given by x.
GPCA.row.gap.filling.option	logical value. Default is TRUE. In case of GPCAvarest2-class objects, If gap.filling contains both NA and finite values in the same row, this row will contains all NA values after GPCA. In this case all row values are generated through auto-regression. If GPCA.row.gap.filling.option all insterted non-NA gap.filling values are repleced before returning the function value. Otherwise, in the rows with NAs all values are re-generated. The option TRUE is not safe in case the gaps are vary long becouse the genereted values is used for subsequent auto-regrossion.
factor.series	factor series used by 'factor.series'
origin	start date for generation. See adddate

Value

a matrix or a data frame object

See Also

getVARmodel

Examples

```
library(RGENERATE)

set.seed(122)
NSTEP <- 1000
x <- rnorm(NSTEP)
y <- x+rnorm(NSTEP)
z <- c(rnorm(1),y[-1]+rnorm(NSTEP-1))
df <- data.frame(x=x,y=y,z=z)
var <- VAR(df,type="none")
gg <- generate(var,n=20)

cov <- cov(gg)

ggg <- generate(FUN=rnorm,n=NSTEP,cov=cov)


library(RMAWGEN)
exogen <- as.data.frame(x+5)
gpcavar <- getVARmodel(data=df,suffix=NULL,p=3,n_GPCA_iteration=5,
n_GPCA_iteration_residuals=5,exogen=exogen)
gpcagg <- generate(gpcavar,n=20,exogen=exogen)

## Generate an auto-regrassive time-series with a generic matrix

A <- diag(c(1,-1,1))
mgg <- generate(A,n=100)

### Gap Filling Examples

 dfobs <- df
 dfobs[20:30,] <- NA
 n <- nrow(df)
 dffill <- generate(gpcavar,n=n,exogen=exogen,gap.filling=dfobs,names=names(dfobs))

qqplot(dfobs$y,dffill$y)
abline(0,1)

### Gap filling with matrix

mgg_n <- mgg
mgg_n[20:30,2] <- NA

mgg_nfill <- generate(A,gap.filling=mgg_n)

print(mgg_n[1:31,])
print(mgg_nfill[1:31,])
```

```
dfobs2 <- df
dfobs2[20:30,2] <- NA
n <- nrow(df)
dffill2 <- generate(gpcavar,n=n,exogen=exogen,gap.filling=dfobs2,names=names(dfobs2))

qqplot(dfobs$y,dffill$y)
abline(0,1)

### generation with 'generetion.matrix'
### and matrix 'x' is a covariance matrix

covariance <- array(0.5,c(3,3))

diag(covariance) <- 1

set.seed(127)
ngns <- 1000
gg1 <- generate(FUN=rnorm,n=ngns,cov=covariance)
set.seed(127)
gg2 <- generate(covariance,type="covariance",n=ngns)


## generate with a list of covariance matrix
ndim <- 5
dim <- c(ndim,ndim)
CS1 <- array(0.3,dim)
CS2 <- array(0.5,dim)
CS3 <- array(0.7,dim)
CS4 <- array(0.1,dim)

diag(CS1) <- 1
diag(CS2) <- 1
diag(CS3) <- 1
diag(CS4) <- 1

list <- list(CS1=CS1,CS2=CS2,CS3=CS3,CS4=CS4)

series <- rep(1:4,times=4,each=100)
series <- sprintf("CS%d",series)
names_A <- sprintf("A%d",1:ndim)
ggs <- generate(list,factor.series=series,FUN=rnorm,type="covariance",names=names_A)

ggs_CS1 <- ggs[series=="CS1",]
cov(ggs_CS1)

ggs_CS3 <- ggs[series=="CS3",]
cov(ggs_CS3)
```

Index