Package 'far'

September 17, 2024

Version 0.6-7

Date 2024-09-13

Title Modelization for Functional AutoRegressive Processes

Depends R (>= 3.4.0), nlme, graphics, stats

Description Modelizations and previsions functions for
Functional AutoRegressive processes using
nonparametric methods: functional kernel,
estimation of the covariance operator in
a subspace, ...

License LGPL-2.1

Encoding UTF-8

URL https://github.com/Looping027/far

NeedsCompilation yes

Author Julien Damon [aut, cre],
Serge, Guillas [aut]

Maintainer Julien Damon <julien.damon@gmail.com>

Repository CRAN

Date/Publication 2024-09-17 12:30:10 UTC

Contents

---

base.simul.far   *Creating functional basis*

Description

Computation of a particular basis in a functional space.

Usage

```
base.simul.far(m=24, n=5)
```

Arguments

m	Number of discretization points
n	Number of axis

Details

We consider a sinusoidal basis of the functional space C[0;1] of the continuous functions from [0;1] to R. We compute here the values of the n first (functional) axis at m equi-repartited discretization points in [0;1] (more precisely the point $0, \frac{1}{m}, ..., \frac{m-1}{m}$).

Value

A matrix of size m x n containing the m values of the n first axis of the basis.

Note

The chosen basis is orthogonal.

The aim of this function is to provide an internal tool for the function simul.farx.

Author(s)

J. Damon

See Also

simul.farx

Examples

```
print(temp<-base.simul.far(10,3))
print(t(temp)%*%temp)
matplot(base.simul.far(100,5),type='l')
```

BaseK2BaseC	*Changing Basis*

Description

Given the coordinates in the Karhunen-Loève expansion base of the Wiener, compute the coordinates in the canonical basis.

Usage

```
BaseK2BaseC(x, nb)
```

Arguments

x	A matrix containing the coordinates in the Karhunen-Loève basis. One observation per column.
nb	The dimension of the canonical basis consider. By default, the dimension is the same as the Karhunen-Loève one (i.e. number of row of x).

Details

The Karhunen-Loève expansion is a sum of an infinity of terms, but here the expansion is truncated to a finite number of terms. Empirically, we remark that using twice the dimension of the canonical basis desired for the number of terms in the expansion is a good compromise.

Value

A object of class fdata with nb discretization points and the same number of observations as x.

Author(s)

J. Damon

References

Pumo, B. (1992). *Estimation et Prévision de Processus Autoregressifs Fonctionnels. Applications aux Processus à Temps Continu*. PhD Thesis, University Paris 6, Pierre et Marie Curie.

See Also

[simul.wiener](#), [simul.far.wiener](#)

Examples

```
data1 <- BaseK2BaseC(x=matrix(rnorm(50),ncol=5,nrow=10), nb=5)
multplot(data1,whole=TRUE)
```

---

coef.far   *Extract Model Coefficients*

Description

'coef' method to extract the linear operator of a FAR model.

Usage

```
## S3 method for class 'far'
coef(object, ...)
```

Arguments

object	An object of type far.
...	Other arguments (not used in this case).

Details

Give the matricial representation of the linear operator express in the canonical basis. See [far](#) for more details about the meaning of this operator.

If the far model is used on a one dimensional variable or with the joined=TRUE option, then the matrix has a dimension equal to the subspace dimension.

In the other case, the dimension of the matrix is equal to the sum of the dimensions of the various subspaces. In such a case, the order of the variables in the matrix is the same as in the vector c(y,x). For instance, if kn=c(3,2) with y="Var1" and x="Var3" then:

- The first 3x3 first bloc of the matrix is the autocorrelation of "Var1".
- The 3x2 up right bloc of the matrix is the correlation of "Var3" on "Var1".
- The 2x3 down left bloc of the matrix is the correlation of "Var1" on "Var3".
- The 2x2 down right bloc of the matrix is the autocorrelation of "Var3".

Value

A square matrix of size (raw and column) equal to the sum of the element of kn.

Author(s)

J. Damon, S. Guillas

See Also

[far](#),[coef](#)

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
            base.exo=base.simul.far(20,5),
            d.a=matrix(c(0.5,0),nrow=1,ncol=2),
            alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
            d.rho=diag(c(0.45,0.90,0.34,0.45)),
            alpha=diag(c(0.5,0.23,0.018)),
            d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
            cst1=0.0)

# Modelization of the FARX process (joined and separate)
model1 <- far(data1,kn=4,joined=TRUE)
model2 <- far(data1,kn=c(3,1),joined=FALSE)

# Calculation of the theoretical coefficients
coef.theo <- theoretical.coef(m=10,base=base.simul.far(20,5),
            base.exo=base.simul.far(20,5),
            d.a=matrix(c(0.5,0),nrow=1,ncol=2),
            alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
            d.rho=diag(c(0.45,0.90,0.34,0.45)),
            alpha=diag(c(0.5,0.23,0.018)),
            d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
            cst1=0.0)

# Joined coefficient
round(coef(model1),2)
coef.theo$rho.T

# Separate coefficient
round(coef(model2),2)
coef.theo$rho.X.Z
```

---

date.fdata   *Extract the date of fdata*

Description

Extract the date(s) of fdata objects

Usage

```
date.fdata(data)
```

Arguments

data   A fdata object

Details

The dates are the labels of the functionals observations of the fdata object.

fdata are not constructed as ts object so a specific function to obtain the date is useful.

Value

A vector giving the dates (as character).

Author(s)

J. Damon

See Also

[fdata](#fdata)

Examples

```
# Reading the data
library(stats)
data(UKDriverDeaths)

# Conversion of the data
fUKDriverDeaths <- as.fdata(UKDriverDeaths,col=1,p=12,dates=1969:1984,
                             name="UK Driver Deaths")
date.fdata(fUKDriverDeaths)
```

---

fapply   *Apply functions over a fdata object*

Description

`fapply` returns a `fdata` object of the same length as data. Each element of which is the result of applying FUN to the corresponding element of data.

Usage

```
fapply(data, FUN, row.names, ...)
```

Arguments

data	A fdata object
FUN	the function to be applied. In the case of functions like +, %*%, etc., the function name must be quoted.
row.names	a vector giving the names describing the results of FUN
...	optional arguments to FUN.

Details

This function has to be used only with `fdata` objects, unless it stop, returning no value.

Value

The returned value is a `fdata` object too.

Author(s)

J. Damon

See Also

[apply](#), [lapply](#).

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
                base.exo=base.simul.far(20,5),
                d.a=matrix(c(0.5,0),nrow=1,ncol=2),
                alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
                d.rho=diag(c(0.45,0.90,0.34,0.45)),
                alpha=diag(c(0.5,0.23,0.018)),
                d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
                cst1=0.0)
```

```
fapply(data1,sum)
multplot(fapply(fapply(data1,abs),cumsum))
```

---

far   *FARX(1) model estimation*

Description

Estimates the parameters of FAR(1) and FARX(1) processes (mean and autocorrelation operator)

Usage

```
far(data, y, x, kn, center=TRUE, na.rm=TRUE, joined=FALSE)
```

Arguments

data	A `fdata` object.
y	A vector giving the name(s) of the endogenous variable(s) of the model.
x	A vector giving the name(s) of the exogenous variable(s) of the model.
kn	A vector giving the values of the various kn (dimension of plug-in in the algorithm). If it not supplied, the default value is one.
center	Logical. Does the observation need to be centered.
na.rm	Logical. Does the `n.a.` need to be removed.
joined	Logical. If `TRUE`, the joined (whole) far model is computed, otherwise the model work with the separated variables.

Details

The models

A Functional AutoRegressive of order 1 (**FAR(1)**) process is, in a general way, defined by the following equation:

$$T_n = \rho\left(T_{n-1}\right) + \epsilon_n, n \in Z$$

where T_n and ϵ_n take their values in a functional space (for instance an Hilbertian one), and ρ is a linear operator. ϵ_n is a strong white noise.

Now, let us consider a vector of observations, for instance:

$$(T_{1,n}, ..., T_{i,n}, ..., T_{m,n})$$

where each $T_{i,n}$ lives in a one dimension functional space (not necessary the same). In the following, we will cut this list into two parts: the endogeneous variables Y_n (the ones we are interested in), and the exogeneous variables X_n (which influence the endogeneous ones).

Then an order 1 Functional AutoRegressive process with eXogeneous variables (**FARX(1)**) is defined by the equation:

$$Y_n = \rho\left(Y_{n-1}\right) + a\left(X_n\right) + \epsilon_n, n \in Z$$

where ρ and a are linear operators in the adequate spaces.

Estimation

This function estimates the parameters of FAR and FARX models.

First, if the mean of the data is not zero (which is required by the model), you can substance this mean using the center option. Moreover, if the data contains NA values, you can work with it using the na.rm option.

FAR Estimation

The estimation is mainly about estimating the ρ operator. This estimation is done in a appropriate subspace (computed from the variance of the observations). What is important to know is that the best dimension kn for this subspace is not determined by this function. So the user have to supply this dimension using the kn option. A way to chose this dimension is to first use the far.cv function on the history.

FARX Estimation

The FARX estimation can be realized by two methods: joined or not.

The **joined** estimation is done by "joining" the variables into one and estimating a FAR model on the resulting variable. For instance, with the previous notations, the transformation is:

$$T_n = (Y_n, X_{n+1})$$

and T_n is then a peculiar FAR(1) process. In such a case, you have to use the joined=TRUE oto the interpretation of this operatorption and specify **one** value for kn (corresponding to the T_n variable).

Alternatively, you can choose not to estimate the FARX model by the joined procedure, then kn need to be a vector with a length equal to the number of variables involved in the FARX model (endogeneous and exogeneous).

In both procedures, the endogeneous and exogeneous variables are provided through the y and x options respectively.

Results

The function returns a far object. Use the print, coef and predict functions to get more informations about the model.

Value

A far object, see details for more informations.

Note

This function could be used to estimate FAR and FARX with order higher than 1 as a change of variables can transform the process to an order 1 FAR or FARX. For instance, if T_n is a FAR(2) process then $Y_n = (T_n, T_{n-1})$ is a FAR(1) process.

However, this is not a basic use of this function and may require a hard work of the user to get the result.

Author(s)

J. Damon

References

Besse, P. and Cardot, H. (1996). Approximation spline de la prévision d'un processus fonctionnel autorégressif d'ordre 1. *Revue Canadienne de Statistique/Canadian Journal of Statistics*, **24**, 467–487.

Bosq, D. (2000) *Linear Processes in Function Spaces: Theory and Applications*, (Lecture Notes in Statistics, Vol. 149). New York: Springer-Verlag.

See Also

predict.far, [far.cv](far.cv)

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
              base.exo=base.simul.far(20,5),
              d.a=matrix(c(0.5,0),nrow=1,ncol=2),
              alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
              d.rho=diag(c(0.45,0.90,0.34,0.45)),
              alpha=diag(c(0.5,0.23,0.018)),
              d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
              cst1=0.0)

# Cross validation (joined and separate)
model1.cv <- far.cv(data=data1, y="X", x="Z", kn=8, ncv=10, cvcrit="X",
              center=FALSE, na.rm=FALSE, joined=TRUE)
model2.cv <- far.cv(data=data1, y="X", x="Z", kn=c(4,4), ncv=10, cvcrit="X",
              center=FALSE, na.rm=FALSE, joined=FALSE)
print(model1.cv)
print(model2.cv)
k1 <- model1.cv$minL2[1]
k2 <- model2.cv$minL2[1:2]

# Modelization of the FARX process (joined and separate)
model1 <- far(data=data1, y="X", x="Z", kn=k1,
              center=FALSE, na.rm=FALSE, joined=TRUE)
model2 <- far(data=data1, y="X", x="Z", kn=k2,
              center=FALSE, na.rm=FALSE, joined=FALSE)
print(model1)
print(model2)
```

---

far.cv   *Cross Validation for FARX(1) model*

Description

Cross Validation for FAR(1) and FARX(1) models

Usage

```
far.cv(data, y, x, kn, ncv, cvcrit, center=TRUE, na.rm=TRUE, joined=FALSE)
```

Arguments

data	A fdata object.
y	A vector giving the name(s) of the endogenous variable(s) of the model.
x	A vector giving the name(s) of the exogenous variable(s) of the model.
kn	A vector giving the maximum values of the various kn (dimension of plug-in in the algorithm). If it not supplied, the number of discretization point is used.
ncv	Number of observations used to the cross validation
cvcrit	A vector of characters. Name of the variable used to measure the errors (y by default).
center	Logical. Does the observation need to be centered.
na.rm	Logical. Does the n.a. need to be removed.
joined	Logical. If TRUE, the joined (whole) far model is computed, otherwise the model work with the separated variables.

Details

In order to perform good forecasting with a FAR or FARX model, you need to determine the dimensions kn of the subspace in which the linear operator is estimated (see [far](#) for more details).

This function helps the user to do this choice by performing a cross validation on a test sample. The usage is close of the [far](#) function, so we will discuss about the options which differ.

First, the kn option is used to restrict the values searched: this is a vector containing the maxima values. As in [far](#), the dimension of this vector is function of the number of variables involved in the model and the type of estimation done (joined or not).

ncv is the number of observation used to test the models. If it is not provided, the function use the last fifth of the observations in data. In such a case, the four first fifth are used to estimates the models. This is in general a good compromise.

Finally, cvcrit list the variables used to test the models. If more than one variable is provided, the test is calculated as a mean of the errors over all the variables.

The criteria used to test the (functional) errors are the norms L1, L2, L infinite, L1 on the maxima, L2 on the maxima, and L infinite on the maxima.

Value

It is a LIST with the following elements

cv	Matrix giving the various errors (L1, L2, L infinite, L1 on the maxima, L2 on the maxima, L infinite on the maxima) for the tested values of kn
minL1	A vector corresponding to the row of cv where the L1 error minima is obtained
minL2	A vector corresponding to the row of cv where the L2 error minima is obtained
minLinf	A vector corresponding to the row of cv where the L infinite error minima is obtained
minL1max	A vector corresponding to the row of cv where the L1 maxima's error minima is obtained
minL2max	A vector corresponding to the row of cv where the L2 maxima's error minima is obtained
minLinfmax	A vector corresponding to the row of cv where the L infinite maxima's error minima is obtained

Author(s)

J. Damon

See Also

[far](), [fdata]()

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
              base.exo=base.simul.far(20,5),
              d.a=matrix(c(0.5,0),nrow=1,ncol=2),
              alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
              d.rho=diag(c(0.45,0.90,0.34,0.45)),
              alpha=diag(c(0.5,0.23,0.018)),
              d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
              cst1=0.0)

# Cross validation (joined and separate)
model1.cv <- far.cv(data=data1, y="X", x="Z", kn=8, ncv=10, cvcrit="X",
              center=FALSE, na.rm=FALSE, joined=TRUE)
model2.cv <- far.cv(data=data1, y="X", x="Z", kn=c(4,4), ncv=10, cvcrit="X",
              center=FALSE, na.rm=FALSE, joined=FALSE)
print(model1.cv)
print(model2.cv)
k1 <- model1.cv$minL2[1]
k2 <- model2.cv$minL2[1:2]

# Modelization of the FARX process (joined and separate)
model1 <- far(data=data1, y="X", x="Z", kn=k1,
              center=FALSE, na.rm=FALSE, joined=TRUE)
```

```
model2 <- far(data=data1, y="X", x="Z", kn=k2,
              center=FALSE, na.rm=FALSE, joined=FALSE)
print(model1)
print(model2)
```

---

fdata   *Functional Data class*

Description

Object of class 'fdata' and its methods.

Usage

```
as.fdata(object,...)
as.fdata.matrix(object,..., col, p, dates, name)
as.fdata.list(object,..., dates, name)
```

Arguments

object	A matrix or a list.
col	A vector giving the names of the variables to include in the 'fdata' object.
p	A real value giving the number of discretization point chosen.
dates	A vector of character containing the dates of the observations.
name	A vector of character containing the names of the variables (generated if not provided).
...	Additional arguments.

Details

Fdata objects are mainly used to modelize functional data in the purpose of computing functional autoregressive model by the [far](#) and [kerfon](#) functions.

An fdata is composed of one or several variables. Each ones is a functional time series.

To be more precise, every variable got a functional data by element of the dates (explicitly given or implicitly deduced). So the number of functional observations is a common data.

In the contrary, each variable can be expressed in a different functional space. For example, if you got two variables, Temperature and Wind, measured during 30 days. Choosing a daily representation, the fdata will contain a 30 elements long dates vector. Nevertheless, the variables measurement can be different. If Temperature is measured every hour and Wind every two hours, the fdata object can handle such a representation. The only constraint is to get a regular measurement: no changes in the methodology.

Basically, the fdata objects are discrete measurements but the modelization which can be used on it will make it functional. Indeed, The first methods implemented as far and kerfon use a linear approximation, but more sophisticate modelization, as splines or wavelets approximations may come.

Value

An object of class fdata.

Author(s)

J. Damon

See Also

[far](), [multplot](), [maxfdata](), [kerfon]().

Examples

```
# Reading of the data
library(stats)
data(UKDriverDeaths)

# Making the data of class 'fdata'
fUKDriverDeaths <- as.fdata(UKDriverDeaths,col=1,p=12,dates=1969:1984,
                            name="UK Driver Deaths")
summary(fUKDriverDeaths)

# ploting of the data : whole and 1 year
par(mfrow=c(2,1))
plot(fUKDriverDeaths,xval=1969+(1:192)/12,whole=TRUE,
     name="Whole Evolution : ")
plot(fUKDriverDeaths,date="1984",xval=1:12,
     name="Evolution during year 1984 : ")

# Matrix conversion
print(as.fdata(matrix(rnorm(50),10,5)))
print(as.fdata(matrix(rnorm(500),100,5),col=1:2,p=5))

# List Conversions
print(as.fdata(list("X"=matrix(rnorm(100),10,10),
"Z"=matrix(rnorm(50),5,10))))
```

---

interpol.matrix   *Interpolation matrix*

Description

Calculate the matrix giving the linear interpolation of regularly spaced points.

Usage

```
interpol.matrix(n = 12, m = 24, tol = sqrt(.Machine$double.eps))
```

Arguments

n	Number (integer) of points in output space
m	Number (integer) of points in the input function (or space)
tol	A relative tolerance to detect zero singular values.

Details

The general principle is, considering a function for which we know values at m equally spaced points (for instance 1/m, 2/m, ..., 1), to compute the matrix giving the linear approximation of n equally spaced points (for instance 1/n, 2/n, ..., 1).

The function works whether n or m is the largest.

The function is vectorized, so m and n can be vectors of integers. In this case, they have to be of the same size and the resulting matrix is block diagonal.

Value

A nxm matrix if they are integer, else a sum(n)xsum(m) matrix.

Author(s)

J. Damon

See Also

[theoretical.coef](), [simul.far]() or [simul.farx]().

Examples

```
mat1 <- interpol.matrix(12,24)
mat2 <- interpol.matrix(c(3,5),c(12,12))
print(mat1 %*% base.simul.far(24,5))
print(mat2 %*% base.simul.far(24,5))
```

---

invgen	*Generalized inverse of a Matrix*

Description

Calculates the Moore-Penrose generalized inverse of a matrix X.

Usage

```
invgen(a, tol)
```

Arguments

a	Matrix for which the Moore-Penrose inverse is required.
tol	A relative tolerance to detect zero singular values.

Value

A Moore-Penrose generalized inverse matrix for X.

See Also

[solve](),[svd](),[eigen]()

Examples

```
mat1<-matrix(rnorm(100),ncol=10)
print(invgen(mat1))
```

---

is.na.fdata   *Not Available / "Missing" Values*

Description

The generic function is.na returns a logical vector of the same "form" as its argument x, containing TRUE for those elements marked NA or NaN (!) and FALSE otherwise. dim, dimnames and names attributes are preserved.

Usage

```
## S3 method for class 'fdata'
is.na(x)
```

Arguments

x	A fdata object

Details

An observation is considered as NA if any of its values is NA.

Value

A matrix of Logical values giving as rows the variables of x and as columns the observation.

Author(s)

J. Damon

See Also

Examples

```
# Reading of the data
library(stats)
data(UKDriverDeaths)
UKDriverDeaths[20]<-NA

# Making the data of class 'fdata'
fUKDriverDeaths <- as.fdata(UKDriverDeaths,col=1,p=12,dates=1969:1984,
                        name="UK Driver Deaths")
summary(fUKDriverDeaths)
is.na(fUKDriverDeaths)
```

---

kerfon	*Functional Kernel estimation*

Description

Modelization of `fdata` using functional kernel.

Usage

```
kerfon(data, x, r, hmin, hmax, na.rm=TRUE)
```

Arguments

data	A `fdata` object.
x	The name of the studied variable.
r	Number of observations used to cross validate the model.
hmin	Minimal value of the bandwidth.
hmax	Maximal value of the bandwidth.
na.rm	Logical. Does the `n.a.` need to be removed.

Details

This function constructs a functional kernel model and performs the estimation of it's bandwidth.

One nonparametric way to deal with the conditional expectation $\rho(x) = \mathbb{E}\left[X_i \,|\, X_{i-1} = x\right]$, where (X_i) is a H-valued process, is to consider a predictor inspired by the classical kernel regression, as in Nadaraja and Watson. This estimator is defined by :

$$\hat{\rho}_{h_n}(x) = \frac{\sum\limits_{i=1}^{n-1} X_{i+1} \cdot K\left(\frac{\|X_i - x\|_H}{h_n}\right)}{\sum\limits_{i=1}^{n-1} K\left(\frac{\|X_i - x\|_H}{h_n}\right)}, x \in H$$

Where K is a kernel, $\|.\|_H$ is the norm in H, and h_n is the bandwidth ($\in \mathbb{R}_*^+$).

The function kerfon use the cross validation to determinate a value for h_n. This method have been chosen because of the lack of theoretical results about this model. The parameters hmin and hmax are used, when provided, to control the permissible values of h_n. By default, those parameters are respectively equals to $\sigma/8$ and $4*\sigma$, where σ is the estimated squared root of the variance operator of X. To choose the value of h_n, you need to provide the same value for both hmin and hmax.

During the cross-validation, considering that the fdata object x contains n observations, the function use the first $(n-r)$ observations as the past values, and compute the mean square norm of the errors on the last r observations.

Of course, if the model created is then used to compute prediction through [predict.kerfon](#), the whole set of observations (the n observations) are used as the past values.

As fdata object may contains several variables, a way is provided to select the studied variable (the function only works with one variable for the moment).

Value

A kerfon object. A method for the [print](#) function is provided.

For information, the object is a list with the following elements :

call	the call of the function.
h	the bandwidth (three values : optimal, minimum, maximum)
x	the name of the chosen variable
xdata	the past values for x
ydata	the associated values for x

Author(s)

J. Damon

See Also

[predict.kerfon](#)

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
              base.exo=base.simul.far(20,5),
              d.a=matrix(c(0.5,0),nrow=1,ncol=2),
              alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
              d.rho=diag(c(0.45,0.90,0.34,0.45)),
              alpha=diag(c(0.5,0.23,0.018)),
              d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
              cst1=0.0)

# Cross validation
model1 <- kerfon(data=data1, x="X", r=10, na.rm=TRUE)
print(model1)
```

---

maxfdata

Maxima of functional data

Description

Extract the maxima series from a functional data object.

Usage

```
maxfdata(data)
```

Arguments

data   A fdata object

Value

A fdata object.

Author(s)

J. Damon

See Also

fapply

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
              base.exo=base.simul.far(20,5),
              d.a=matrix(c(0.5,0),nrow=1,ncol=2),
              alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
              d.rho=diag(c(0.45,0.90,0.34,0.45)),
              alpha=diag(c(0.5,0.23,0.018)),
              d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
              cst1=0.0)

print(data2 <- maxfdata(data1))
print(unclass(data2))
```

---

multplot   *Multivariate plots*

Description

Multivariate plots of Functional Data (more precisely fdata objects).

Usage

```
multplot(object, ...)

## S3 method for class 'fdata'
 multplot(object, date = 1, xval = NULL, name = NULL, legend = FALSE,
     yleg, xlab = NULL, ylab = NULL, main = NULL, whole = FALSE, ...)
```

Arguments

object	An fdata object for which a multplot is desired.
date	String vector. List of the dates to work with.
xval	Numerical vector. Values of the axis x.
name	String vector. The set of variables to plot.
legend	Boolean. Plot a legend ?
yleg	Numeric. Where to put the legend box (y value).
xlab	String. Title of the axis x.
ylab	String. Title of the axis y.
main	String. Title of the plot.
whole	Boolean. A global plot (TRUE) or a plot by day (FALSE)
...	Additional arguments.

Details

This function facilitate the plotting of fdata objects. It is dedicated to multivariate plots, please take a look at [plot.fdata](#) if you need univariate plots in one graphic.

The default behaviour is to produce one plot containing all the variables of the observation called "1".

If you want less variables, use the name argument. If you need more observations, use the date argument. When provided, the xval argument allow you to change the labels of the x-axis.

It is also possible to plot the complete series on the same plot using the whole argument.

Moreover a legend facility is provided using the legend and yleg arguments.

Author(s)

J. Damon

See Also

[fdata](), [plot.fdata]().

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=100,base=base.simul.far(20,5),
             base.exo=base.simul.far(20,5),
             d.a=matrix(c(0.5,0),nrow=1,ncol=2),
             alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
             d.rho=diag(c(0.45,0.90,0.34,0.45)),
             alpha=diag(c(0.5,0.23,0.018)),
             d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
             cst1=0.0)

# 2 variables : X et Z
# number of points per curve : 10
# number of curves : 100
# corresponding dates
date.fdata(data1)

multplot(data1) # plot the date "1" of the variables "X" and "Z"
multplot(data1,legend=TRUE) # Same thing with a legend
multplot(data1,legend=TRUE,yleg=-0.5) # same thing with a legend misplaced
multplot(data1,main="day 1",legend=TRUE,xlab="hour",
        ylab="object of study")

par(mfrow=c(1,3))
multplot(data1,date=c("3","4","5")) # days "3", "4" and "5" are plotted
par(mfrow=c(1,1))

# to plot the whole series, we used whole = TRUE
# but we have to give the x values

multplot(data1,xval=seq(from=0,to=99.9,by=0.1),whole=TRUE)

# to plot a subset of the series,
# it is recommended to create a subset object with select.fdata
data2 <- select.fdata(data1,date=c("4","5","6"))
multplot(data2,xval=seq(from=4,to=6.9,by=0.1),whole=TRUE)
```

---

orthonormalization   *Orthonormalization of a set of a matrix*

Description

Gram-Schmidt orthogonalization of a matrix considering its columns as vectors. Normalization is provided as will.

Usage

```
orthonormalization(u, basis=TRUE, norm=TRUE)
```

Arguments

u   a matrix (n x p) representing n different vectors in a n dimensional space

basis   does the returned matrix have to be a basis

norm   does the returned vectors have to be normed

Details

This is a simple application of the Gram-Schmidt algorithm of orthogonalization (please note that this process was presented first by Laplace).

The user provides a set of vector (structured in a matrix) and the function calculate a orthogonal basis of the same space. If desired, the returned basis can be normed, or/and completed to cover the hole space.

If the number of vectors in u is greater than the dimension of the space (that is if $n > p$), only the first p columns are taken into account to computed the result. A warning is also provided.

The only assumption made on u is that the span space is of size min(n,p). In other words, there must be no colinearities in the initial set of vector.

Value

The orthogonalized matrix obtained from u where the vector are arranged in columns.

If basis is set to TRUE, the returned matrix is squared.

Author(s)

J. Damon

Examples

```
mat1 <- matrix(c(1,0,1,1,1,0),nrow=3,ncol=2)
orth1 <- orthonormalization(mat1, basis=FALSE, norm=FALSE)
orth2 <- orthonormalization(mat1, basis=FALSE, norm=TRUE)
orth3 <- orthonormalization(mat1, basis=TRUE, norm=TRUE)
crossprod(orth1)
crossprod(orth2)
crossprod(orth3)
```

---

plot.fdata *Plot Functional Data*

Description

Plot Functional Data (more precisely fdata objects).

Usage

```
## S3 method for class 'fdata'
plot(x,...,date, xval, name, main, whole, separator)
```

Arguments

x	A fdata object.
date	A vector of character giving the chosen dates.
xval	A numerical vector giving the values to appear on the x axis.
name	A vector of character giving the plotted variables.
main	an overall title for the plot.
whole	Logical. If TRUE all the observations are plot on the same graphic.
separator	Logical. It will be used when whole=TRUE. If TRUE then dashed lines are plotted to separated the observations.
...	Additional arguments to the plot.

Details

This function facilitate the plotting of fdata objects. It is dedicated to univariate plots, please take a look at multplot if you need multivariate plots in one graphic.

The default behaviour is to plot the observation called "1" of all the variables available in x (so it will produce as many plots as the number of variables).

If you want less variables, use the name argument. If you need more observations, use the date argument. When provided, the xval argument allow you to change the labels of the x-axis.

It is also possible to plot the complete series on the same plot using the whole argument. In this case, the separator allow you to draw line to distinguish the different observations of the functional data.

Author(s)

J. Damon

See Also

fdata, multplot.

Examples

```
# Reading of the data
library(stats)
data(UKDriverDeaths)

# Making the data of class 'fdata'
fUKDriverDeaths <- as.fdata(UKDriverDeaths, col=1, p=12,
                            dates=1969:1984,
                            name="UK Driver Deaths")
summary(fUKDriverDeaths)

# plotting of the data : whole and 1 year
par(mfrow=c(2,1))
plot(fUKDriverDeaths, xval=1969+(1:192)/12,
     whole=TRUE, name="Whole Evolution : ", separator=TRUE)
plot(fUKDriverDeaths, date="1984", xval=1:12,
     name="Evolution during year 1984 : ")
```

---

pred.persist   *Forecasting using functional persistence*

Description

Compute prediction of functional data using the persistence.

Usage

```
pred.persist(data, x, na.rm=TRUE, label, positive=FALSE)
```

Arguments

data	A fdata object.
x	A vector of character giving the names of the variables predicted.
na.rm	Logical. Does the n.a. need to be removed.
label	A vector of character giving the dates to associate to the predicted observations.
positive	Logical. Does the result must be forced to positive values.

Details

The persistence model is a beautiful way to name the simplest model ever. This model just suppose that the next observation will be equal to the previous one, that is to say, noting $\hat{X}_n$ the prediction for X_n that we "compute" :

$$\hat{X}_{n+1} = X_n$$

Of course, the intrinsic purpose of this model is to be a comparison for more complicated models.

The x option is provided to select the variable to predict, using the `label` option value as the labels for the new observations. Notices that the output as the same length as the input as it is only a shift in time.

In some special context, the user may need to suppress the `na.rm` observations with the `na.rm` option, or force the prediction to be positive with the `positive` option (in this case the maximum of 0 and the past value is computed).

Value

A `fdata` object.

Note

This has been more instinctive to call this function predict.persist but, due to the naming mechanism introduced by the object oriented programming, this would have reefer to the predict method for the persist objects. As it isn't the meaning of this function, we preferred the name pred.persist.

Author(s)

J. Damon

See Also

predict.far,predict.kerfon.

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=40,base=base.simul.far(20,5),
                base.exo=base.simul.far(20,5),
                d.a=matrix(c(0.5,0),nrow=1,ncol=2),
                alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
                d.rho=diag(c(0.45,0.90,0.34,0.45)),
                alpha=diag(c(0.5,0.23,0.018)),
                d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
                cst1=0.0)
print(data2 <- pred.persist(data1,x="X",label="41"))
print(unclass(select.fdata(data1,date=paste(38:40)))$X)
print(unclass(select.fdata(data2,date=paste(39:41))))
```

---

predict.far	*Forecasting of FARX(1) model*

Description

Forecasting using FAR(1) or FARX(1) model

Usage

```
## S3 method for class 'far'
predict(object, ..., newdata=NULL, label, na.rm=TRUE, positive=FALSE)
```

Arguments

object	A far object result of the far function.
newdata	A data matrix (one column for each observation) used to predict the FAR(1) model from the values in newdata, or NULL to predict one step forward with the data in object.
label	A vector of character giving the dates to associate to the predicted observations.
na.rm	Logical. Does the n.a. need to be removed.
positive	Logical. Does the result must be forced to positive values.
...	Additional arguments.

Details

This function computes one step forward prediction for a [far](#) model.

Use the newdata option to input the past values, and the label option value to define the labels for the new observations. Notices that the output as the same length as newdata in the case of a FAR model, and the length of newdata minus one in the case of a FARX model. This is due to the time shift of the exogeneous variable: X_{t+1} and Y_t are used in the computation of $\hat{Y}_{t+1}$.

In some special context, the user may need to suppress the na.rm observations with the na.rm option, or force the prediction to be positive with the positive option (in this case the result will be maximum of 0 and the predicted value).

Value

A fdata object.

Author(s)

J. Damon

See Also

[far](#), [pred.persist](#), [predict.kerfon](#).

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
                base.exo=base.simul.far(20,5),
                d.a=matrix(c(0.5,0),nrow=1,ncol=2),
                alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
                d.rho=diag(c(0.45,0.90,0.34,0.45)),
                alpha=diag(c(0.5,0.23,0.018)),
                d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
```

```
                    cst1=0.0)

    # Cross validation (joined and separate)
    model1.cv <- far.cv(data=data1, y="X", x="Z", kn=8, ncv=10, cvcrit="X",
                   center=FALSE, na.rm=FALSE, joined=TRUE)
    model2.cv <- far.cv(data=data1, y="X", x="Z", kn=c(4,4), ncv=10, cvcrit="X",
                   center=FALSE, na.rm=FALSE, joined=FALSE)
    print(model1.cv)
    print(model2.cv)
    k1 <- model1.cv$minL2[1]
    k2 <- model2.cv$minL2[1:2]

    # Modelization of the FARX process (joined and separate)
    model1 <- far(data=data1, y="X", x="Z", kn=k1,
                   center=FALSE, na.rm=FALSE, joined=TRUE)
    model2 <- far(data=data1, y="X", x="Z", kn=k2,
                   center=FALSE, na.rm=FALSE, joined=FALSE)

    # Predicting values
    pred1 <- predict(model1,newdata=data1)
    pred2 <- predict(model2,newdata=data1)
    # Persistence
    persist1 <- pred.persist(select.fdata(data1,date=1:399),x="X")
    # Real values
    real1 <- select.fdata(data1,date=2:400)

    errors0 <- persist1[[1]]-real1[[1]]
    errors1 <- pred1[[1]]-real1[[1]]
    errors2 <- pred2[[1]]-real1[[1]]

    # Norm of observations
    summary(real1)
    # Persistence
    summary(as.fdata(errors0))
    # FARX models
    summary(as.fdata(errors1))
    summary(as.fdata(errors2))
```

---

predict.kerfon   *Forecasting of functional kernel model*

Description

Computation of the prediction based on a functional kernel model

Usage

```
## S3 method for class 'kerfon'
predict(object, ..., newdata=NULL, label, na.rm=TRUE, positive=FALSE)
```

Arguments

object	A kerfon object result of the kerfon function.
newdata	A [fdata](#) object used in the kerfon model to compute the prediction, or NULL to predict one step forward with the data in object.
label	A vector of character giving the dates to associate to the predicted observations.
na.rm	Logical. Does the n.a. need to be removed.
positive	Logical. Does the result must be forced to positive values.
...	Additional arguments.

Details

This function computes one step forward prediction for a [kerfon](#) model.

Use the newdata option to input the past values, and the label option value to define the labels for the new observations. Notices that the output as the same length as newdata.

In some special context, the user may need to suppress the na.rm observations with the na.rm option, or force the prediction to be positive with the positive option (in this case the result will be maximum of 0 and the predicted value).

Value

A fdata object.

Author(s)

J. Damon

See Also

[kerfon](#)

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
             base.exo=base.simul.far(20,5),
             d.a=matrix(c(0.5,0),nrow=1,ncol=2),
             alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
             d.rho=diag(c(0.45,0.90,0.34,0.45)),
             alpha=diag(c(0.5,0.23,0.018)),
             d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
             cst1=0.0)

# Cross validation
model1 <- kerfon(data=data1, x="X", r=10, na.rm=TRUE)
print(model1)

# Predicting values
pred1 <- predict(model1,newdata=select.fdata(data1,date=1:399))
```

```
# Persistence
persist1 <- pred.persist(select.fdata(data1,date=1:399),x="X")
# Real values
real1 <- select.fdata(data1,date=2:400)

errors0 <- persist1[[1]]-real1[[1]]
errors1 <- pred1[[1]]-real1[[1]]

# Norm of observations
summary(real1)
# Persistence
summary(as.fdata(errors0))
# kerfon model
summary(as.fdata(errors1))
```

---

select.fdata   *Subscript of fdata*

Description

Use this function to subscript some functional observations of a functional data.

Usage

```
select.fdata(data, date, name)
```

Arguments

data	A fdata object.
date	A vector of character containing the chosen dates (could be NULL).
name	A vector giving the chosen name (could be NULL).

Details

This function select one or several variables from data and can also subset the dates. This is useful in order to study the endogenous variables of a FARX process.

Value

A fdata object.

Author(s)

J. Damon

See Also

[fdata](#)

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
              base.exo=base.simul.far(20,5),
              d.a=matrix(c(0.5,0),nrow=1,ncol=2),
              alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
              d.rho=diag(c(0.45,0.90,0.34,0.45)),
              alpha=diag(c(0.5,0.23,0.018)),
              d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
              cst1=0.0)

print(data1)
print(data1.X <- select.fdata(data1,name="X"))
print(data2 <- select.fdata(data1,date=paste((1:5)*5)))
date.fdata(data2)
```

---

simul.far   *FAR(1) process simulation*

Description

Simulation of a FAR process using a Gram-Schmidt basis.

Usage

```
simul.far(m=12,
  n=100,
  base=base.simul.far(24, 5),
  d.rho=diag(c(0.45, 0.9, 0.34, 0.45)),
  alpha=diag(c(0.5, 0.23, 0.018)),
  cst1=0.05)
```

Arguments

m	Integer. Number of discretization points.
n	Integer. Number of observations.
base	A functional basis expressed as a matrix, as the matrix created by `base.simul.far` or with `orthonormalization`.
d.rho	Numerical matrix. Expression of the first bloc of the linear operator in the Gram-Schmidt basis.
alpha	Numerical matrix. Expression of the first bloc of the covariance operator in the Gram-Schmidt basis.
cst1	Numeric. Perturbation coefficient on the linear operator.

Details

This function simulate a FAR(1) process with a strong white noise.

The simulation is realized in two steps.

First step, the function compute a FAR(1) process T_n in a functional space (that we call in the sequel H) using a simple equation and the `d.rho`, `alpha` and `cst` parameters.

Second step, the process T_n is projected in the canonical basis using the `base` linear projector.

The `base` basis need to be a orthonormal basis wide enought. In the contrary, the function use the [orthonormalization](#) function to make it so. Notice that the size of this matrix corresponds to the dimension of the "modelization space" H (let's call it m_2). Of course, the larger `m2` the better the functionnal approximation is. Whatever, keep in mind that `m2=2m` is a good compromise, in order to avoid the memory limits.

In H, the linear operator ρ is expressed as:

$$\begin{pmatrix} \texttt{d.rho} & 0 \\ 0 & eps.rho \end{pmatrix}$$

Where `d.rho` is the matrix provided in the call, the two 0 are in fact two blocks of 0, and eps.rho is a diagonal matrix having on his diagonal the terms:

$$\left(\varepsilon_{k+1}, \varepsilon_{k+2}, \ldots, \varepsilon_{\texttt{m2}} \right)$$

where

$$\varepsilon_i = \frac{\texttt{cst1}}{i^2} + \frac{1 - \texttt{cst1}}{e^i}$$

and `k` is the length of the `d.rho` diagonal.

The `d.rho` matrix can be viewed as the information and the eps.rho matrix as a perturbation. In this logic, the norm of eps.rho need to be smaller than the one of `d.rho`.

In H, C^T, the covariance operator of T_n, is defined by:

$$\begin{pmatrix} m_2 * \texttt{alpha} & 0 \\ 0 & eps.alpha \end{pmatrix}$$

Where `alpha` is the matrix provided in the call, the two 0 are in fact two blocks of 0, and eps.alpha is a diagonal matrix having on his diagonal the terms:

$$\left(\epsilon_{k+1}, \epsilon_{k+2}, \ldots, \epsilon_{\texttt{m2}} \right)$$

where

$$\epsilon_i = \frac{\texttt{cst1}}{i}$$

Value

A `fdata` object containing one variable ("var") which is a FAR(1) process of length `n` with `p` discretization points.

Note

To simulate T_n, the function creates a white noise E_n having the following covariance operator:

$$C^T - \rho * C^T * t(\rho)$$

where t(.) is the transposition operator. T_n is the computed using the equation:

$$T_{n+1} = \rho * T_n + E_n$$

Author(s)

J. Damon, S. Guillas

See Also

simul.far.sde, simul.far.wiener, simul.farx, simul.wiener, base.simul.far.

Examples

```
far1 <- simul.far(m=64,n=100)
summary(far1)
print(far(far1,kn=4))
par(mfrow=c(2,1))
plot(far1,date=1)
plot(select.fdata(far1,date=1:5),whole=TRUE,separator=TRUE)
```

---

simul.far.sde   *FAR-SDE process simulation*

Description

Simulation of a FAR process following an Stochastic Differential Equation

Usage

```
simul.far.sde(coef=c(0.4, 0.8), n=80, p=32, sigma=1)
```

Arguments

coef	Numerical vertor. It contains the two values of the coefficients (a_1 and a_2, see details for more informations).
n	Integer. The number of observations generated.
p	Integer. The number of discretization points.
sigma	Numeric. The standard deviation (see details for more informations).

Details

This function implements the simulation proposed by Besse and Cardot (1996) to simulate a FAR process following the Stochastic Differential Equation:

$$dX^{(2)} + a_2.dX + a_1.X = \texttt{sigma}.dW$$

Where $dX^{(2)}$ and dX stand respectively for the second and first derivate of the process X, and W is a brownian process.

The coefficients a_1 and a_2 are the two first elements of `coef`.

The simulation use a order one approximation inspired by the work of Milstein, as described in Besse and Cardot (1996).

Value

A `fdata` object containing one variable ("var") which is a FAR(1) process of length `n` with `p` discretization points.

Author(s)

J. Damon

References

Besse, P. and Cardot, H. (1996). *Approximation spline de la prévision d'un processus fonctionnel autorégressif d'ordre 1.* Revue Canadienne de Statistique/Canadian Journal of Statistics, **24**, 467–487.

See Also

`simul.far`, `simul.far.wiener`, `simul.farx`, `simul.wiener`.

Examples

```
far1 <- simul.far.sde()
summary(far1)
print(far(far1,kn=2))
par(mfrow=c(2,1))
plot(far1,date=1)
plot(select.fdata(far1,date=1:5),whole=TRUE,separator=TRUE)
```

---

simul.far.wiener   *FAR(1) process simulation with Wiener noise*

Description

Simulation of a FAR(1) process using a Wiener noise.

Usage

```
simul.far.wiener(m=64, n=128,
d.rho=diag(c(0.45, 0.9, 0.34, 0.45)), cst1=0.05, m2=NULL)
```

Arguments

m	Integer. Number of discretization points.
n	Integer. Number of observations.
d.rho	Numerical matrix. Expression of the first bloc of the linear operator in the Karhunen-Loève basis.
cst1	Numeric. Perturbation coefficient on the linear operator.
m2	Integer. Length of the Karhunen-Loève expansion (2m by default).

Details

This function simulate a FAR(1) process with a Wiener noise. As for the `simul.wiener`, the function use the Karhunen-Loève expansion of the noise. The FAR(1) process, defined by its linear operator (see `far` for more details), is computed in the Karhunen-Loève basis then projected in the natural basis. The parameters given in input (`d.rho` and `cst1`) are expressed in the Karhunen-Loève basis.

The linear operator, expressed in the Karhunen-Loève basis, is of the form:

$$\left(\begin{array}{cc} \texttt{d.rho} & 0 \\ 0 & eps.rho \end{array} \right)$$

Where `d.rho` is the matrix provided in ths call, the two 0 are in fact two blocks of 0, and eps.rho is a diagonal matrix having on his diagonal the terms:

$$\left(\varepsilon_{k+1}, \varepsilon_{k+2}, \ldots, \varepsilon_{\texttt{m2}} \right)$$

where

$$\varepsilon_i = \frac{\texttt{cst1}}{i^2} + \frac{1 - \texttt{cst1}}{e^i}$$

and k is the length of the `d.rho` diagonal.

The `d.rho` matrix can be viewed as the information and the eps.rho matrix as a perturbation. In this logic, the norm of eps.rho need to be smaller than the one of `d.rho`.

Value

A `fdata` object containing one variable ("var") which is a FAR(1) process of length n with m discretization points.

Author(s)

J. Damon

References

Pumo, B. (1992). *Estimation et Prévision de Processus Autoregressifs Fonctionnels. Applications aux Processus à Temps Continu.* PhD Thesis, University Paris 6, Pierre et Marie Curie.

See Also

[fdata](), [far]() , [simul.far.wiener]().

Examples

```
far1 <- simul.far.wiener(m=64,n=100)
summary(far1)
print(far(far1,kn=4))
par(mfrow=c(2,1))
plot(far1,date=1)
plot(select.fdata(far1,date=1:5),whole=TRUE,separator=TRUE)
```

---

simul.farx	*FARX(1) process simulation*

Description

Simulation of functional data with exogenous variables using a Gram-Schmidt basis.

Usage

```
simul.farx(m=12,n=100,base=base.simul.far(24,5),
  base.exo=base.simul.far(24,5),
  d.a=matrix(c(0.5,0),nrow=1,ncol=2),
  alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
  d.rho=diag(c(0.45,0.90,0.34,0.45)),
  alpha=diag(c(0.5,0.23,0.018)),
  d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
  cst1=0.05)
theoretical.coef(m=12,base=base.simul.far(24,5),
  base.exo=NULL,
  d.rho=diag(c(0.45,0.90,0.34,0.45)),
  d.a=NULL,
  d.rho.exo=NULL,
```

```
alpha=diag(c(0.5,0.23,0.018)),
alpha.conj=NULL,
cst1=0.05)
```

Arguments

m	Integer. Number of discretization points.
n	Integer. Number of observations.
base	A functional basis expressed as a matrix, as the matrix created by base.simul.far or with orthonormalization.
base.exo	A functional basis expressed as a matrix, as the matrix created by base.simul.far or with orthonormalization.
d.rho	Numerical matrix. Part of the linear operator in the Gram-Schmidt basis (see details for more informations).
d.a	Numerical matrix. Part of the linear operator in the Gram-Schmidt basis (see details for more informations).
d.rho.exo	Numerical matrix. Part of the linear operator in the Gram-Schmidt basis (see details for more informations).
alpha	Numerical matrix. Part of the linear operator in the Gram-Schmidt basis (see details for more informations).
alpha.conj	Numerical matrix. Part of the linear operator in the Gram-Schmidt basis (see details for more informations).
cst1	Numeric. Perturbation coefficient on the linear operator.

Details

The simul.farx function simulates a FARX(1) process with one endogeneous variable, one exogeneous variable and a strong white noise. To do so, the function uses the fact that a FARX(1) model can be seen as a FAR(1) model in a wider space. Therefore, the method is very similar to the one used by the function simul.far.

The simulation is realized in two steps.

First step, the function compute a FAR(1) process T_n in a functional space (that we call in the sequel H) using a simple equation and the given parameters. T_n is of the form (T_{1n}, T_{2n}) where T_{1n} and T_{2n} are respectively the endogeneous and the exogeneous parts of the process.

Second step, the process T_n is projected in the canonical basis using the base and base.exo linear projectors to give the endogeneous (X_n) and the exogeneous (Z_n) variables respectively.

Those two basis need to be orthonormal and wide enought. In the contrary, the function use the orthonormalization function to make it so. Notice that the size of this matrix corresponds to the dimension of the "modelization space" H (let's call it $m_2 = m1_2 + m2_2$). Of course, the larger m2 the better the functionnal approximation is. Whatever, keep in mind that m2=2m is a good compromise, in order to avoid the memory limits.

In H, the linear operator ρ is expressed as:

$$\begin{pmatrix} d.rho.mod & \text{d.a} \\ 0 & d.rho.exo.mod \end{pmatrix}$$

Where d.rho.mod and d.rho.exo.mod are modified version of the provided `d.rho` and `d.rho.exo` respectively to avoid 0 on their diagonal. More precisely, the 0 on their diaginals are replaced by:

$$(\varepsilon_{k+1}, \varepsilon_{k+2}, \ldots, \varepsilon_{\mathtt{m2}})$$

where

$$\varepsilon_i = \frac{\mathtt{cst1}}{i^2} + \frac{1 - \mathtt{cst1}}{e^i}$$

and `k` is the position in the `d.rho` or `d.r.ho.exo` diagonal.

In H, C^T, the covariance operator of T_n, is defined by:

$$\begin{pmatrix} alpha.mod & alpha.conj.mod \\ t(alpha.conj.mod) & alpha.exo \end{pmatrix}$$

Where alpha.mod and alpha.exo.mod are modified versions of $m1_2*$`alpha` and $m2_2*$`alpha.conj` respectively to avoid 0 on their diagonal. More precisely, the 0 on their diaginals are replaced by:

$$(\epsilon_{k+1}, \epsilon_{k+2}, \ldots, \epsilon_{\mathtt{m2b}})$$

where

$$\epsilon_i = \frac{\mathtt{cst1}}{i}$$

alpha.exo is a matrix representation of the covariance operator of T_{2n} and is obtained by inverting the following relation:

$$alpha.conj.mod = d.rho.exo.mod*alpha.conj.mod*t(d.rho.mod) + d.rho.exo.mod*mod.alpha*t(\mathtt{d.a})$$

The `theoretical.coef` function is provided to help the user making comparison. Calling this function with the same parameters that where used in a simulation (realized with `simul.farx` or `simul.far`), we obtain the parameters used internaly by the function to make the simulation. Those values can therefore be compared to those obtained with the estimation function `far` (examples are provided below).

Value

A `fdata` object containing two variables ("X" the endogeous variable, and "Z" the exogeneous variable) which is a FARX(1) process of length `n` with `p` discretization points.

Note

To simulate T_n, the function creates a white noise E_n having the following covariance operator:

$$C^T - \rho * C^T * t(\rho)$$

where t(.) is the transposition operator. T_n is the computed using the equation:

$$T_{n+1} = \rho * T_n + E_n$$

Author(s)

J. Damon, S. Guillas

See Also

[simul.far.sde](), [simul.far.wiener](), [simul.far](), [simul.wiener]().

Examples

```
# Simulation of a FARX process
data1 <- simul.farx(m=10,n=400,base=base.simul.far(20,5),
                base.exo=base.simul.far(20,5),
                d.a=matrix(c(0.5,0),nrow=1,ncol=2),
                alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
                d.rho=diag(c(0.45,0.90,0.34,0.45)),
                alpha=diag(c(0.5,0.23,0.018)),
                d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
                cst1=0.0)

# Modelisation of the FARX process (joined and separate)
model1 <- far(data1,k=4,joined=TRUE)
model2 <- far(data1,k=c(3,1),joined=FALSE)

# Calculation of the theoretical coefficients
coef.theo <- theoretical.coef(m=10,base=base.simul.far(20,5),
                base.exo=base.simul.far(20,5),
                d.a=matrix(c(0.5,0),nrow=1,ncol=2),
                alpha.conj=matrix(c(0.2,0),nrow=1,ncol=2),
                d.rho=diag(c(0.45,0.90,0.34,0.45)),
                alpha=diag(c(0.5,0.23,0.018)),
                d.rho.exo=diag(c(0.45,0.90,0.34,0.45)),
                cst1=0.0)

# Joined coefficient
round(coef(model1),2)
coef.theo$rho.T

# Separate coefficient
round(coef(model2),2)
coef.theo$rho.X.Z
```

---

simul.wiener   *Wiener process simulation*

Description

Simulation of Wiener processes.

Usage

```
simul.wiener(m=64, n=1, m2=NULL)
```

Arguments

m	Integer. Number of discretization points.
n	Integer. Number of observations.
m2	Integer. Length of the Karhunen-Loève expansion (2m by default).

Details

This function use the known Karhunen-Loève expansion of Wiener processes to simulate observations of such a process.

The option `m2` is internally used to set the length of the expansion. This expansion need to be larger than the number of discretization points, but a too important value may slow down the generation. The default value as been chosen as a compromise.

Value

A `fdata` object containing one variable ("var") which is a Wiener process of length `n` with `m` discretization points.

Author(s)

J. Damon

References

Pumo, B. (1992). *Estimation et Prévision de Processus Autoregressifs Fonctionnels. Applications aux Processus à Temps Continu.* PhD Thesis, University Paris 6, Pierre et Marie Curie.

See Also

[simul.far.sde](), [simul.far.wiener](), [simul.farx](), [simul.far]().

Examples

```
noise <- simul.wiener(m=64,n=100,m2=512)
summary(noise)
par(mfrow=c(2,1))
plot(noise,date=1)
plot(select.fdata(noise,date=1:5),whole=TRUE,separator=TRUE)
```

Index