Package 'Rsfar'

May 10, 2021

Type Package

Title Seasonal Functional Autoregressive Models

Version 0.0.1

Description This is a collection of functions designed for simulating, estimating and forecasting seasonal functional autoregressive time series of order one. These methods are addressed in the manuscript: <https://www.monash.edu/business/ebs/research/publications/ebs/wp16-2019.pdf>.

License GPL (>= 2)

URL https://github.com/haghbinh/Rsfar

Encoding UTF-8

LazyData false

RoxygenNote 7.1.1

Depends fda

NeedsCompilation no

Author Hossein Haghbin [aut, cre] (<https://orcid.org/0000-0001-8416-2354>), Rob Hyndman [aut]

Maintainer Hossein Haghbin <haghbinh@gmail.com>

Repository CRAN

Date/Publication 2021-05-10 08:02:27 UTC

R topics documented:

Rsfar-package *Rsfar: A Package for Seasonal Functional Autoregressive Models.*

Description

The Rsfar package provides the collection of necessary functions for simulating, estimating and forecasting seasonal functional autoregressive time series of order one.

Details

Functional autoregressive models are popular for functional time series analysis, but the standard formulation fails to address seasonal behavior in functional time series data. To overcome this shortcoming, we introduce seasonal functional autoregressive time series models. For the model of order one, we provide estimation, prediction and simulation methods.

References

Atefeh Z., Hossein H., Maryam H., and R.J Hyndman (2021). Seasonal functional autoregressive models. Manuscript submitted for publication. https://robjhyndman.com/publications/sfar/

See Also

sfar, predict.sfar,

Bdiag *Create block diagonal matrix*

Description

Create block diagonal matrix

Usage

```
Bdiag(A, k)
```

Arguments

A	a numeric matrix forming each block.
k	an integer value indicating the number of blocks.

Value

Return a block diagonal matrix from the matrix A.

Examples

```
Bdiag(matrix(1:4,2,2), 3)
```

invsqrt *Return inverse square root of square positive-definite matrix.*

Description

Return inverse square root of square positive-definite matrix.

Usage

```
invsqrt(A)
```

Arguments

A	a numeric matrix

Value

inverse square root of square positive-definite matrix.

Examples

```
require(Rsfar)
X <- Bdiag(matrix(1:4,2,2), 3)
invsqrt(t(X) %*% X)
```

predict.sfar *Prediction of an SFAR model*

Description

Compute h-step-ahead prediction for an SFAR(1) model. Only the h-step predicted function is returned, not the predictions for 1,2,...,h.

Usage

```
## S3 method for class 'sfar'
predict(object, h, ...)
```

Arguments

object	an ‘sfar‘ object containing a fitted SFAR(1) model.
h	number of steps ahead to predict.
...	Other parameters, not currently used.

Value

An object of class fda.

Examples

```
# Generate Brownian motion noise
N <- 300 # the length of the series
n <- 200 # the sample rate that each function will be sampled
u <- seq(0, 1, length.out = n) # argvalues of the functions
d <- 45 # the number of bases
basis <- create.fourier.basis(c(0, 1), d) # the basis system
sigma <- 0.05 # the std of noise norm
Z0 <- matrix(rnorm(N * n, 0, sigma), nrow = n, nc = N)
Z0[, 1] <- 0
Z_mat <- apply(Z0, 2, cumsum) # N standard Brownian motion
Z <- smooth.basis(u, Z_mat, basis)$fd

# Simulate random SFAR(1) data
kr <- function(x, y) {
  (2 - (2 * x - 1)^2 - (2 * y - 1)^2) / 2
}
s <- 5 # the period number
X <- rsfar(kr, s, Z)
plot(X)

# SFAR(1) model parameter estimation:
Model1 <- sfar(X, seasonal = s, kn = 1)

# Forecasting 3 steps ahead
fc <- predict(Model1, h = 3)
plot(fc)
```

---

rsfar *Simulation of a Seasonal Functional Autoregressive SFAR(1) process.*

Description

Simulation of a SFAR(1) process on a Hilbert space of L2[0,1].

Usage

```
rsfar(phi, seasonal, Z)
```

Arguments

phi	a kernel function corresponding to the seasonal autoregressive operator.
seasonal	a positive integer variable specifying the seasonal period.
Z	the functional noise object of the class 'fd'.

Value

A sample of functional time series from a SFAR(1) model of the class ‘fd‘.

Examples

```
# Set up Brownian motion noise process
N <- 300 # the length of the series
n <- 200 # the sample rate that each function will be sampled
u <- seq(0, 1, length.out = n) # argvalues of the functions
d <- 15 # the number of basis functions
basis <- create.fourier.basis(c(0, 1), d) # the basis system
sigma <- 0.05 # the stdev of noise norm
Z0 <- matrix(rnorm(N * n, 0, sigma), nr = n, nc = N)
Z0[, 1] <- 0
Z_mat <- apply(Z0, 2, cumsum) # N standard Brownian motion
Z <- smooth.basis(u, Z_mat, basis)$fd

# Compute the standardized constant of a kernel function with respect to a given HS norm.
gamma0 <- function(norm, kr) {
  f <- function(x) {
    g <- function(y) {
      kr(x, y)^2
    }
    return(integrate(g, 0, 1)$value)
  }
  f <- Vectorize(f)
  A <- integrate(f, 0, 1)$value
  return(norm / A)
}
# Definition of parabolic integral kernel:
norm <- 0.99
kr <- function(x, y) {
  2 - (2 * x - 1)^2 - (2 * y - 1)^2
}
c0 <- gamma0(norm, kr)
phi <- function(x, y) {
  c0 * kr(x, y)
}

# Simulating a path from an SFAR(1) process
s <- 5 # the period number
X <- rsfar(phi, s, Z)
plot(X)
```

`sfar` *Estimation of an SFAR(1) Model*

Description

Estimate a seasonal functional autoregressive (SFAR) model of order 1 for a given functional time series.

Usage

```
sfar(
  X,
  seasonal,
  cpv = 0.85,
  kn = NULL,
  method = c("MME", "ULSE", "KOE"),
  a = ncol(Coefs)^(-1/6)
)
```

Arguments

X	a functional time series.
seasonal	a positive integer variable specifying the seasonality parameter.
cpv	a numeric with values in [0,1] which determines the cumulative proportion variance explained by the first kn eigencomponents.
kn	an integer variable specifying the number of eigencomponents.
method	a character string giving the method of estimation. The following values are possible: "MME" for Method of Moments, "ULSE" for Unconditional Least Square Estimation Method, and "KOE" for Kargin-Ontaski Estimation.
a	a numeric with value in [0,1].

Value

A matrix of size p*p.

Examples

```
# Generate Brownian motion noise
N <- 300 # the length of the series
n <- 200 # the sample rate that each function will be sampled
u <- seq(0, 1, length.out = n) # argvalues of the functions
d <- 45 # the number of bases
basis <- create.fourier.basis(c(0, 1), d) # the basis system
sigma <- 0.05 # the std of noise norm
Z0 <- matrix(rnorm(N * n, 0, sigma), nrow = n, nc = N)
Z0[, 1] <- 0
Z_mat <- apply(Z0, 2, cumsum) # N standard Brownian motion
Z <- smooth.basis(u, Z_mat, basis)$fd

# Simulate random SFAR(1) data
kr <- function(x, y) {
 (2 - (2 * x - 1)^2 - (2 * y - 1)^2) / 2
}
s <- 5 # the period number
X <- rsfar(kr, s, Z)
plot(X)

# SFAR(1) model parameter estimation:
```

```
Model1 <- sfar(X, seasonal = s, kn = 1)
```

Index