Package 'DCSmooth'

October 12, 2022

Type Package

Title Nonparametric Regression and Bandwidth Selection for Spatial Models

Version 1.1.2

Author Bastian Schaefer [aut, cre],
Sebastian Letmathe [ctb],
Yuanhua Feng [ths]

Maintainer Bastian Schaefer <bastian.schaefer@uni-paderborn.de>

Description Nonparametric smoothing techniques for data on a lattice and functional time series. Smoothing is done via kernel regression or local polynomial regression, a bandwidth selection procedure based on an iterative plug-in algorithm is implemented. This package allows for modeling a dependency structure of the error terms of the nonparametric regression model. Methods used in this paper are described in Feng/Schaefer (2021) <https://ideas.repec.org/p/pdn/ciepap/144.html>, Schaefer/Feng (2021) <https://ideas.repec.org/p/pdn/ciepap/143.html>.

License GPL-3

Depends R (>= 3.1.0)

Imports doParallel, foreach, fracdiff, parallel, plotly, Rcpp, stats

Suggests knitr, rmarkdown, testthat

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-10-21 15:20:08 UTC

R topics documented:

`DCSmooth-package` *Nonparametric Regression and Bandwidth Selection for Spatial Models*

Description

Nonparametric smoothing techniques for data on a lattice and functional time series. Smoothing is done via kernel regression or local polynomial regression, a bandwidth selection procedure based on an iterative plug-in algorithm is implemented. This package allows for modeling a dependency structure of the error terms of the nonparametric regression model. Methods used in this paper are described in Feng/Schaefer (2021) <https://ideas.repec.org/p/pdn/ciepap/144.html>, Schaefer/Feng (2021) <https://ideas.repec.org/p/pdn/ciepap/143.html>.

Package Content

Index of help topics:

DCSmooth-package	Nonparametric Regression and Bandwidth Selection for Spatial Models
dcs	Nonparametric Double Conditional Smoothing for 2D Surfaces
kernel.assign	Assign a Kernel Function
kernel.list	Print a list of available kernels in the DCSmooth package
plot.dcs	Contour Plot for the Double Conditional Smoothing
print.dcs	Summarize Results from Double Conditional Smoothing
print.dcs_options	Print and Summarize Options for Double Conditional Smoothing
print.summary_dcs	Print the Summary of a DCS estimation
print.summary_sarma	Print the Summary of a "sarma"/"sfarima" object
residuals.dcs	Residuals of "dcs"-object
returns.alv	Returns of Allianz SE
sarma.est	Estimation of an SARMA-process
sarma.sim	Simulation of a SARMA(p, q)-process
set.options	Set Options for the DCS procedure
sfarima.est	Estimation of a SFARIMA-process
sfarima.sim	Simulation of a SFARIMA(p, q, d)-process
summary.dcs	Summarizing Results from Double Conditional Smoothing
summary.dcs_options	Print and Summarize Options for Double Conditional Smoothing
summary.sarma	Summarizing SARMA/SFARIMA Estimation or Simulation
surface.dcs	3D Surface Plot of "dcs"-object or numeric matrix
temp.nunn	Temperatures from Nunn, CO
temp.yuma	Temperatures from Yuma, AZ
volumes.alv	Volumes of Allianz SE
wind.nunn	Wind Speed from Nunn, CO
wind.yuma	Wind Speed from Yuma, AZ
y.norm1	Single Gaussian Peak
y.norm2	Double Gaussian Peak
y.norm3	Double Gaussian Ridges

Maintainer

Bastian Schaefer <bastian.schaefer@uni-paderborn.de>

Author(s)

Bastian Schaefer [aut, cre], Sebastian Letmathe [ctb], Yuanhua Feng [ths]

dcs *Nonparametric Double Conditional Smoothing for 2D Surfaces*

Description

dcs provides a double conditional nonparametric smoothing of the expectation surface of a functional time series or a random field on a lattice. Bandwidth selection is done via an iterative plug-in method.

Usage

```
dcs(Y, dcs_options = set.options(), h = "auto", parallel = FALSE, ...)
```

Arguments

Y	A numeric matrix that contains the observations of the random field or functional time-series.
dcs_options	An object of class "dcs_options", specifying the parameters for the smoothing and bandwidth selection procedure.
h	Bandwidth for smoothing the observations in Y. Can be a two-valued numerical vector with bandwidths in row- and column-direction. If the value is "auto" (the default), bandwidth selection will be carried out by the iterative plug-in algorithm.
parallel	A logical value indicating if parallel computing should be used for faster computation. Default value is parallel = FALSE. Parallelization seems to be efficient at above 400,000 observations.
...	Additional arguments passed to dcs. Currently supported are numerical vectors X and/or T containing the exogenous covariates with respect to the rows and columns.

Value

dcs returns an object of class "dcs", including

Y	matrix of original observations.
X, T	vectors of covariates over rows (X) and columns (T).
M	resulting matrix of smoothed values.
R	matrix of residuals of estimation, $Y - M$.
h	optimized or given bandwidths.
c_f	estimated variance coefficient.
var_est	estimated variance model. If the variance function is modeled by an SARMA/SFARIMA, var_est is an obje
dcs_options	an object of class cds_options containing the initial options of the dcs procedure.
iterations	number of iterations of the IPI-procedure.
time_used	time spend searching for optimal bandwidths (not overall runtime of the function).

Details

See the vignette for a more detailed description of the function.

References

Schäfer, B. and Feng, Y. (2021). Fast Computation and Bandwidth Selection Algorithms for Smoothing Functional Time Series. Working Papers CIE 143, Paderborn University.

See Also

set.options

Examples

```
# See vignette("DCSmooth") for examples and explanation

y <- y.norm1 + matrix(rnorm(101^2), nrow = 101, ncol = 101)
dcs(y)
```

---

kernel.assign *Assign a Kernel Function*

Description

Assign a Kernel Function

Usage

```
kernel.assign(kernel_id)
```

Arguments

kernel_id	a string specifying the kernel identifier as given in the details.

Value

kernel.assign returns an object of class "function". This function takes two arguments, a numeric vector in the first argument and a single number in the second. The function itself will return a matrix with one column and the same number of rows as the input vector.

Details

kernel.assign sets a pointer to a specified kernel function available in the DCSmooth package. The kernels are boundary kernels of the form $K(u, q)$, where $u \in [-1, q]$ and $q \in [0, 1]$ q = [0, 1]. Kernels are of the Müller-Wang type ("MW"), Müller type ("M") or truncated kernels ("TR").

References

Müller, H.-G. and Wang, J.-L. (1994). Hazard rate estimation under random censoring with varying kernels and bandwidths. Biometrics, 50:61-76.

Müller, H.-G. (1991). Smooth optimum kernel estimators near endpoints. Biometrika, 78:521-530.

Feng, Y. and Schäfer B. (2021). Boundary Modification in Local Regression. Working Papers CIE 144, Paderborn University.

See Also

kernel.list

Examples

```
 # See vignette("DCSmooth") for further examples and explanation

u <- seq(from = -1, to = 0.5, length.out = 151)
kern_MW220 <- kernel.assign("MW_220")
k <- kern_MW220(u, 0.5)
plot(u, k, type = "l")
```

---

kernel.list *Print a list of available kernels in the DCSmooth package*

Description

Print a list of available kernels in the DCSmooth package

Usage

```
kernel.list(print = TRUE)
```

Arguments

print	Logical value. Should the list be printed to the console? If TRUE (the default), the list is printed to the console, if FALSE the list of identifiers is returned from the function as (surprise!) a list.

Value

If print = FALSE, a list is returned containing the kernel identifiers

Details

kernel.list is used to get a list of available kernels in the DCSmooth package.

kernel.list prints a list of identifiers kernel_id of available kernels in the DCSmooth package. The available kernel types are "T": truncated, "MW": Müller-Wang boundary correction, "M": Müller boundary correction.

References

Müller, H.-G. and Wang, J.-L. (1994). Hazard rate estimation under random censoring with varying kernels and bandwidths. Biometrics, 50:61-76.

Müller, H.-G. (1991). Smooth optimum kernel estimators near endpoints. Biometrika, 78:521-530.

Feng, Y. and Schäfer B. (2021). Boundary Modification in Local Regression. Working Papers CIE 144, Paderborn University.

See Also

kernel.assign

Examples

```
# See vignette("DCSmooth") for further examples and explanation

kernel.list()
```

---

plot.dcs *Contour Plot for the Double Conditional Smoothing*

Description

plot method for class "dcs"

Usage

```
## S3 method for class 'dcs'
plot(x, ...)
```

Arguments

x	an object of class "dcs_options", usually, a result of a call to set.options.
...	Additional arguments passed to print.dcs_options. The argument plot_choice overrides the prompt to specify a plot, can be c(1, 2, 3).

Value

No return value.

Details

plot.dcs provides a contour plot of either the original data (1), smoothed surface (2) or residuals (3).

See Also

surface.dcs to plot the surface.

Examples

```
## Contour plot of smoothed surface
y <- y.norm1 + matrix(rnorm(101^2), nrow = 101, ncol = 101)
dcs_object <- dcs(y)
plot(dcs_object, plot_choice = 2)
```

`print.dcs` *Summarize Results from Double Conditional Smoothing*

Description

`print` method for class `"dcs"`

Usage

```
## S3 method for class 'dcs'
print(x, ...)
```

Arguments

`x`	an object of class "dcs", usually, a result of a call to `dcs`.
`...`	Additional arguments passed to `print.dcs`.

Value

No return value.

Details

`print.dcs` prints a short summary of an object of class `dcs`, only including bandwidths and the estimated variance coefficient (only if automatic bandwidth selection is used).

See Also

`plot.dcs`, `print.dcs_options`

Examples

```
y <- y.norm1 + matrix(rnorm(101^2), nrow = 101, ncol = 101)
dcs_object <- dcs(y)
print(dcs_object)
dcs_object
```

print.dcs_options *Print and Summarize Options for Double Conditional Smoothing*

Description

print method for class "dcs_options"

Usage

```
## S3 method for class 'dcs_options'
print(x, ...)
```

Arguments

x	an object of class "dcs_options", usually, a result of a call to set.options.
...	Additional arguments passed to print.dcs_options.

Value

No return value.

Details

print.dcs_options prints the main options and summary.dcs_options prints main and advanced (IPI) options used for the dcs function. Arguments should be an object of class "dcs_options".

See Also

print.dcs, summary.dcs_options

Examples

```
## Default options
myOpt <- set.options()
print(myOpt)
summary(myOpt)

## Use Kernel regression
myOpt <- set.options(type = "KR")
print(myOpt)
summary(myOpt)
```

print.summary_dcs *Print the Summary of a DCS estimation*

Description

print method for class "summary_dcs"

Usage

```
## S3 method for class 'summary_dcs'
print(x, ...)
```

Arguments

x	An object of class "summary_dcs".
...	Additional arguments passed to print.summary_dcs.

Value

No return value.

See Also

summary.dcs

print.summary_sarma *Print the Summary of a "sarma"/"sfarima" object*

Description

print methods for class "summary_sarma"/ "summary_sfarima"

Usage

```
## S3 method for class 'summary_sarma'
print(x, ...)

## S3 method for class 'summary_sfarima'
print(x, ...)
```

Arguments

x	An object of class "summary_sarma" or "summary_sfarima".
...	Additional arguments passed to print.summary_sarma/ print.summary_sfarima.

Value

No return value.

See Also

summary.sarma summary.sfarima

`residuals.dcs` *Residuals of "dcs"-object*

Description

Returns the residuals of an object of class "dcs".

Usage

```
## S3 method for class 'dcs'
residuals(x, ...)
```

Arguments

x	an object of class "dcs", usually the result of a call to dcs.
...	Additional arguments passed to residuals.dcs.

Value

Returns the $n_x \times n_t$-matrix of residuals.

See Also

dcs

Examples

```
y = y.norm1 + matrix(rnorm(101^2), nrow = 101, ncol = 101)
dcs_object = dcs(y)
residuals(dcs_object)
```

returns.alv *Returns of Allianz SE*

Description

The (log-) returns of the shares of the German insurance company Allianz SE from 2007-01-02 to 2010-12-30 aggregated to 5-minute observations. The data is adjusted to matrix form for direct use with the DCSmooth-functions.

Usage

```
returns.alv
```

Format

A numeric matrix with 1016 rows representing the days and 101 columns representing the intraday time points.

sarma.est *Estimation of an SARMA-process*

Description

Parametric Estimation of an $SARMA(p, q)$-process on a lattice.

Usage

```
sarma.est(Y, method = "HR", model_order = list(ar = c(1, 1), ma = c(1, 1)))

qarma.est(Y, model_order = list(ar = c(1, 1), ma = c(1, 1)))
```

Arguments

Y	A numeric matrix that contains the demeaned observations of the random field or functional time-series.
method	Method used for estimation of the parameters. One of "HR", "sep", "RSS", default value is "HR"
model_order	A list containing the orders of the SARMA model in the form model_order = list(ar = c(p1, p2), ma = c(q1, q2)). Default value is a $SARMA((1,1),(1,1))$ model.

Value

The function returns an object of class "sarma" including

Y	The matrix of observations, inherited from input.
innov	The estimated innovations.
model	The estimated model consisting of the coefficient matrices ar and ma and standard deviati
stnry	An logical variable indicating whether the estimated model is stationary.

Details

The MA- and AR-parameters of a top-left quadrant ARMA process are estimated by the specified method. The lag-orders of the $SARMA(p, q)$ are given by $p = (p_1, p_2), q = (q_1, q_2)$, where p_1, q_1 are the lags over the rows and p_2, q_2 are the lags over the columns. The estimation process is based on the model

$$\phi(B_1 B_2) X_{i,j} = \theta(B_1 B_2) u_{i,j}$$

.

See Also

sarma.sim, sfarima.est

Examples

```
# See vignette("DCSmooth") for examples and explanation

## simulation of SARMA process
ma <- matrix(c(1, 0.2, 0.4, 0.1), nrow = 2, ncol = 2)
ar <- matrix(c(1, 0.5, -0.1, 0.1), nrow = 2, ncol = 2)
sigma <- 0.5
sarma_model <- list(ar = ar, ma = ma, sigma = sigma)
sarma_simulated <- sarma.sim(100, 100, model = sarma_model)
sarma_simulated$model

## estimation of SARMA process
sarma.est(sarma_simulated$Y)$model
sarma.est(sarma_simulated$Y,
          model_order = list(ar = c(1, 1), ma = c(1, 1)))$model
```

---

sarma.sim *Simulation of a $SARMA(p, q)$-process*

Description

sarma.sim simulates a specified SARMA-model on a lattice with normally distributed innovations.

Usage

```
sarma.sim(n_x, n_t, model)

qarma.sim(n_x, n_t, model)
```

Arguments

n_x	Number of simulated observation rows.
n_t	Number of simulated observation columns.
model	A list containing the coefficient matrices ar and ma of the SARMA model as well as the standard deviation of innovations sigma.

Value

The function returns an object of class "sarma", consisting of

Y	A $n_x \times n_t$-matrix of simulated values of the specified SARMA process.
innov	The innovations used for simulation, iid. drawn from a normal distribution with zero mean and variance σ^2.
model	The model used for simulation, inherited from input.
stnry	An logical variable indicating whether the simulated model is stationary.

Details

Simulation of a top-left dependent spatial ARMA process (SARMA). This function returns an object of class "sarma". The simulated innovations are created from a normal distribution with specified variance σ^2.

see the vignette for further details.

See Also

sarma.est, sfarima.est

Examples

```
# See vignette("DCSmooth") for examples and explanation

ma <- matrix(c(1, 0.2, 0.4, 0.1), nrow = 2, ncol = 2)
ar <- matrix(c(1, 0.5, -0.1, 0.1), nrow = 2, ncol = 2)
sigma <- 0.5
sarma_model <- list(ar = ar, ma = ma, sigma = sigma)

sarma_sim <- sarma.sim(100, 100, model = sarma_model)
summary(sarma_sim)
```

set.options *Set Options for the DCS procedure*

Description

Set Options for the DCS procedure

Usage

```
set.options(
  type = "LP",
  kerns = c("MW_220", "MW_220"),
  drv = c(0, 0),
  var_model = "iid",
  ...
)
```

Arguments

type	either local polynomial regression ("LP", the default) or kernel regression ("KR").
kerns	a character vector of length 2 containing the identifier for the kernels to be used in kernel regression. Weighting functions in local polynomial regression are computed according to the identifier. Default value is MW_220, the Mueller-Wang kernel of order $(2, 2, 0)$. If only a single value is provided, it is used as kernel in both directions.
drv	A non-negative vector of length 2, containing the derivative orders to be estimated from the given data. The default is c(0, 0). For LP-regression, polynomial order is selected as $(\nu_1 + 1, \nu_2 + 1)$. If only a single value is provided, it is used as derivative in both directions.
var_model	the method of estimating the variance coefficient c_f. Currently available are var_model = c("iid", "sarma_HR", "sarma_sep","sarma_RSS", "sfarima_RSS"). Replacing the argument var_model. For code using var_est, the argument is converted to var_model.
...	Additional arguments passed to set.options(). This includes IPI_options, a list containing further options used by the iterative plug-in algorithm. For convenience, any of the options usually included in the list IPI_options can be passed as argument directly to set.options and will be converted into the IPI_options list. Further arguments accepted are model_order controlling the order of the variance model, if either an SARMA or SFARIMA model is used. This argument is either a list of the form list(ar = c(1, 1), ma = c(1, 1)) or specifies an order selection criterion from c("aic", "bic", "gpac"). If an order selection criterion is used, the argument order_max controls the maximum order to be tested.

Value

An object of class "dcs_options".

Details

This function is used to set the options for bandwidth selection in the dcs function. Detailed information can be found in the vignette.

See Also

dcs

Examples

```
# See vignette("DCSmooth") for examples and explanation

set.options()

myOpt <- set.options(type = "KR", var_model = "iid")
y <- y.norm1 + matrix(rnorm(101^2), nrow = 101, ncol = 101)
dcs(y, dcs_options = myOpt)
```

---

sfarima.est *Estimation of a SFARIMA-process*

Description

Parametric Estimation of a $SFARIMA(p, q, d)$-process on a lattice.

Usage

```
sfarima.est(Y, model_order = list(ar = c(1, 1), ma = c(1, 1)))
```

Arguments

Y	A numeric matrix that contains the demeaned observations of the random field or functional time-series.
model_order	A list containing the orders of the SFARIMA model in the form model_order = list(ar = c(p1, p2), ma = c(q1, q2)). Default value is a $SFARIMA((1,1),(1,1),d)$ model.

Value

The function returns an object of class "sfarima" including

Y	The matrix of observations, inherited from input.
innov	The estimated innovations.
model	The estimated model consisting of the coefficient matrices ar and ma, the estimated long r
stnry	An logical variable indicating whether the estimated model is stationary.

Details

The MA- and AR-parameters as well as the long-memory parameters

$$d$$

of a SFARIMA process are estimated by minimization of the residual sum of squares RSS. Lag-orders of $SFARIMA(p,q,d)$ are given by $p = (p_1, p_2), q = (q_1, q_2)$, where p_1, q_1 are the lags over the rows and p_2, q_2 are the lags over the columns. The estimated process is based on the (separable) model

$$\varepsilon_{ij} = \Psi_1(B)\Psi_2(B)\eta_{ij}$$

, where

$$\Psi_i = (1 - B_i)^{-d_i}\phi_i^{-1}(B_i)\psi_i(B_i), i = 1, 2$$

.

See Also

sarma.est, sfarima.sim

Examples

```
# See vignette("DCSmooth") for examples and explanation

## simulation of SFARIMA process
ma <- matrix(c(1, 0.2, 0.4, 0.1), nrow = 2, ncol = 2)
ar <- matrix(c(1, 0.5, -0.1, 0.1), nrow = 2, ncol = 2)
d <- c(0.1, 0.1)
sigma <- 0.5
sfarima_model <- list(ar = ar, ma = ma, d = d, sigma = sigma)
sfarima_sim <- sfarima.sim(50, 50, model = sfarima_model)

## estimation of SFARIMA process
sfarima.est(sfarima_sim$Y)$model
sfarima.est(sfarima_sim$Y,
           model_order = list(ar = c(1, 1), ma = c(0, 0)))$model
```

`sfarima.sim` *Simulation of a $SFARIMA(p,q,d)$-process*

Description

`sfarima.sim` simulates a specified SFARIMA-model on a lattice with normally distributed innovations.

Usage

```
sfarima.sim(n_x, n_t, model)
```

Arguments

n_x	Number of simulated observation rows.
n_t	Number of simulated observation columns.
model	A list containing the coefficient matrices ar and ma of the QARMA model, the long memory parameter vector d as well as the standard deviation of innovations sigma.

Value

The function returns an object of class "sfarima", consisting of

Y	A $n_x \times n_t$-matrix of simulated values of the specified SFARIMA process.
innov	The innovations used for simulation, iid. drawn from a normal distribution with zero mean and variance σ^2.
model	The model used for simulation, inherited from input.
stnry	An logical variable indicating whether the simulated model is stationary.

Details

Simulation of a separable spatial fractionally ARIMA process (SFARIMA). This function returns an object of class "sfarima". The simulated innovations are created from a normal distribution with specified variance σ^2.

see the vignette for further details.

See Also

qarma.est

Examples

```
# See vignette("DCSmooth") for examples and explanation

ma <- matrix(c(1, 0.2, 0.4, 0.1), nrow = 2, ncol = 2)
ar <- matrix(c(1, 0.5, -0.1, 0.1), nrow = 2, ncol = 2)
d <- c(0.1, 0.1)
sigma <- 0.5
sfarima_model <- list(ar = ar, ma = ma, d = d, sigma = sigma)

sfarima_sim <- sfarima.sim(100, 100, model = sfarima_model)
surface.dcs(sfarima_sim$Y)
```

---

summary.dcs *Summarizing Results from Double Conditional Smoothing*

Description

summary method for class "dcs"

Usage

```
## S3 method for class 'dcs'
summary(object, ...)
```

Arguments

object	an object of class "dcs", usually, a result of a call to dcs.
...	Additional arguments passed to the summary.dcs function.

Value

The function summary.dcs returns an object of class summary_dcs including

h_opt	estimated optimal bandwidth from the IPI-procedure.
c_f	estimated variance factor.
iterations	number of iterations of the IPI-procedure.
time_used	time spend searching for optimal bandwidths (not overall runtime of the function).
var_est	estimated variance model. Has class "sarma" if an SARMA model is used and class "sfarima" if an SFAI
var_model_id	identifier for the variance model estimated.
var_model_order	order of the estimated variance model, if either SARMA or SFARIMA is used.
dcs_options	an object of class cds_options containing the initial options of the dcs procedure.

Details

summary.dcs strips an object of class "dcs" from all large matrices (Y, X, T, M, R), allowing for easier handling of meta-statistics of the bandwidth selection procedure.

print.summary_dcs returns a list of summary statistics from the dcs procedure. The output depends on the use of the dcs- function. If automatic bandwidth selection is chosen, summary.dcs prints detailed statistics of the type of regression, the estimated bandwidths h_x, h_t, the variance coefficient c_f and performance statistics such as the number of iterations of the IPI-algorithm and the time used for bandwidth selection.

The method used for estimation of the variance coefficient is printed and the results of an SARMA/SFARIMA-estimation, if available.

If bandwidths are supplied to dcs, summary.dcs only prints the given bandwidths.

Examples

```
y <- y.norm1 + matrix(rnorm(101^2), nrow = 101, ncol = 101)
dcs_object <- dcs(y)
summary(dcs_object)
```

summary.dcs_options *Print and Summarize Options for Double Conditional Smoothing*

Description

summary method for class "dcs_options"

Usage

```
## S3 method for class 'dcs_options'
summary(object, ...)
```

Arguments

object	an object of class "dcs_options", usually, a result of a call to set.options.
...	Additional arguments passed to summary.dcs_options.

Value

No return value.

Details

print.dcs_options prints the main options and summary.dcs_options prints main and advanced (IPI) options used for the dcs function. Arguments should be an object of class "dcs_options".

See Also

print.dcs, print.dcs_options

Examples

```
## Default options
myOpt <- set.options()
print(myOpt)
summary(myOpt)

## Use Kernel regression
myOpt <- set.options(type = "KR")
print(myOpt)
summary(myOpt)
```

summary.sarma *Summarizing SARMA/SFARIMA Estimation or Simulation*

Description

summary method for class "sarma" or "sfarima"

Usage

```
## S3 method for class 'sarma'
summary(object, ...)

## S3 method for class 'sfarima'
summary(object, ...)
```

Arguments

object	an object of class "sarma" or "sfarima", usually a result of a call to the estimation functions sarma.est, sfarima.est or to the corresponding simulation functions sarma.sim and sfarima.sim.
...	Additional arguments passed to the summary.sarma/ summary.sfarima function.

Value

The function summary.sarma/summary.sfarima returns an object of class summary_sarma including

model	estimated or simulated model parameters including coefficient matrices ar, ma, the error term standard deviati
model_order	order of the estimated/simulated model computed from the matrices ar, ma.
stnry	a flag for stationarity of the short memory part.
subclass	a flag indicating whether the object inherits from an estimation (subclass = "est") or simulation procedure (

Details

summary.sarma/summary.sfarima strips an object of class "sarma"/"sfarima" from all large matrices (Y, innov), allowing for easier handling of meta-statistics of the bandwidth selection procedure.

print.summary_sarma/print.summary_sarma returns a list of summary statistics from the estimation or simulation procedure.

See Also

sarma.est, sfarima.est, sarma.sim, sfarima.sim

Examples

```
# SARMA Simulation and Estimation
ma = matrix(c(1, 0.2, 0.4, 0.1), nrow = 2, ncol = 2)
ar = matrix(c(1, 0.5, -0.1, 0.1), nrow = 2, ncol = 2)
sigma = 0.5
sarma_model = list(ar = ar, ma = ma, sigma = sigma)
sarma_sim = sarma.sim(100, 100, model = sarma_model)
summary(sarma_sim)
sarma_est = sarma.est(sarma_sim$Y)
summary(sarma_est)

# SFARIMA Simulation and Estimation
ma = matrix(c(1, 0.2, 0.4, 0.1), nrow = 2, ncol = 2)
ar = matrix(c(1, 0.5, -0.1, 0.1), nrow = 2, ncol = 2)
d = c(0.1, 0.1)
sigma = 0.5
sfarima_model = list(ar = ar, ma = ma, d = d, sigma = sigma)
sfarima_sim = sfarima.sim(100, 100, model = sfarima_model)
summary(sfarima_sim)
sfarima_est = sfarima.est(sfarima_sim$Y)
summary(sfarima_est)
```

---

surface.dcs *3D Surface Plot of "dcs"-object or numeric matrix*

Description

3D Surface Plot of "dcs"-object or numeric matrix

Usage

```
surface.dcs(Y, trim = c(0, 0), plot_choice = "choice", ...)
```

Arguments

Y	an object of class "dcs" or a numeric matrix that contains the values to be plotted.
trim	a numeric vector with two values specifying the percentage of trimming applied to the boundaries of the surface to plot. Useful for derivative estimation.
plot_choice	override the prompt to specify a plot, can be c(1, 2, 3).
...	optional arguments passed to the plot function.

Value

dcs.3d returns an object of class "plotly" and "htmlwidget".

Details

surface.dcs uses the plotly device to plot the 3D surface of the given "dcs"-object or matrix. If a "dcs"-object is passed to the function, it can be chosen between plots of the original data (1), smoothed surface (2) and residuals (3).

See Also

plot.dcs

Examples

```
# See vignette("DCSmooth") for examples and explanation

smth <-  dcs(y.norm1 + rnorm(101^2))
surface.dcs(smth, trim = c(0.05, 0.05), plot_choice = 2)
```

---

temp.nunn *Temperatures from Nunn, CO*

Description

This dataset contains the 5-minute observations of the 2020 temperature in Nunn, CO. The data is from the U.S. Climate Reference Network database at www.ncdc.noaa.gov. (see Diamond et al., 2013). The observations were adjusted matrix form for direct use with the DCSmooth-functions.

Usage

```
temp.nunn
```

Format

A numeric matrix with 366 rows and 288 columns containing the temperatures in Celsius.

---

temp.yuma *Temperatures from Yuma, AZ*

Description

This dataset contains the 5-minute observations of the 2020 temperature in Yuma, AZ. The data is from the U.S. Climate Reference Network database at www.ncdc.noaa.gov. (see Diamond et al., 2013). The observations were adjusted matrix form for direct use with the DCSmooth-functions.

Usage

```
temp.yuma
```

Format

A numeric matrix with 366 rows and 288 columns containing the temperatures in Celsius.

`volumes.alv` *Volumes of Allianz SE*

Description

The trading volumes of the shares of the German insurance company Allianz SE from 2007-01-02 to 2010-09-30 aggregated to 5-minute observations. The data is adjusted to matrix form for direct use with the `DCSmooth`-functions.

Usage

```
volumes.alv
```

Format

A numeric matrix with 1016 rows representing the days and 102 columns representing the intraday time points.

`wind.nunn` *Wind Speed from Nunn, CO*

Description

This dataset contains the 5-minute observations of the 2020 wind speed in Nunn, CO. The data is from the U.S. Climate Reference Network database at www.ncdc.noaa.gov. (see Diamond et al., 2013). The observations were adjusted matrix form for direct use with the `DCSmooth`-functions.

Usage

```
wind.nunn
```

Format

A numeric matrix with 366 rows and 288 columns containing the wind speed in m/s.

wind.yuma *Wind Speed from Yuma, AZ*

Description

This dataset contains the 5-minute observations of the 2020 wind speed in Yuma, AZ. The data is from the U.S. Climate Reference Network database at www.ncdc.noaa.gov. (see Diamond et al., 2013). The observations were adjusted matrix form for direct use with the DCSmooth-functions.

Usage

```
wind.yuma
```

Format

A numeric matrix with 366 rows and 288 columns containing the wind speeds in m/s.

y.norm1 *Single Gaussian Peak*

Description

Example data for using the DCSmooth functions. Data resembles a single gaussian peak on the interval $[0, 1] \times [0, 1]$ with maximum at $(0.5, 0.5)$ and variance matrix $0.1 \cdot \mathbf{I}$, where $\mathbf{I}$ represents the 2×2 identity matrix.

Usage

```
y.norm1
```

Format

A numeric matrix with 101 rows and 101 columns.

y.norm2 *Double Gaussian Peak*

Description

Example data for using the DCSmooth functions. Data resembles two gaussian peaks on the interval $[0,1] \times [0,1]$ with maxima at $(0.5, 0.3)$ with variance matrix $0.1 \cdot \mathbf{I}$ and at $(0.2, 0.8)$ with variance matrix $0.05 \cdot \mathbf{I}$, where $\mathbf{I}$ represents the 2×2 identity matrix.

Usage

```
y.norm2
```

Format

A numeric matrix with 101 rows and 101 columns.

y.norm3 *Double Gaussian Ridges*

Description

Example data for using the DCSmooth functions. Data resembles two gaussian ridges on the interval $[0,1] \times [0,1]$ with maxima at $(0.25, 0.75)$ with variance matrix $(0.01, -0.1) \cdot \mathbf{I}$ and at $(0.75, 0.5)$ with variance matrix $(0.01, -0.1) \cdot \mathbf{I}$, where $\mathbf{I}$ represents the 2×2 identity matrix.

Usage

```
y.norm3
```

Format

A numeric matrix with 101 rows and 101 columns.

Index