Package 'gpDDE'

December 16, 2015

Title General Profiling Method for Delay Differential Equation

Version 0.8.2

Description Functions implement collocation-inference for stochastic process driven by distributed delay differential equations. They also provide tools for selecting the lags for distributed delay using shrinkage methods, estimating time-varying coefficients, and tools for inference and prediction.

Depends R (>= 3.2.1), fda, CollocInfer (>= 1.0.2)

License GPL (>= 2)

LazyData true

Imports penalized, nnls, deSolve, MASS, TSA, lars, limSolve, forecast, trustOptim, methods

NeedsCompilation no

Author Ziqian Zhou [aut, cre], Kung-Sik Chan [aut]

Maintainer Ziqian Zhou <ziqian-zhou@uiowa.edu>

Repository CRAN

Date/Publication 2015-12-16 23:25:34

R topics documented:

`blowflydata` *Nicholson's Blowflies Data.*

Description

A set of control experiments in which the adult food supply was restricted to 0.4g per day. The adult blowfly population exhibits oscillation due to constant resource level.

Usage

```
blowflydata
```

Format

A data frame with two variables: `day` and `y`. `day` indicates the time observations are made. The experiment last from the 40th day to the 315th day. Observations are made daily. `y` is the blowfly population count.

`DDEdiag` *Time Series Diagnostics for the Residuals*

Description

A function to plot time-series diagnostics of the residuals from a generalized profiling DDE model.

Usage

```
DDEdiag(y, times, fitted, use.TSA = FALSE, ...)
```

Arguments

`y`	Matrix of observed data values.
`times`	Vector observation times for the data.
`fitted`	The functional data object for the estimated state process.
`use.TSA`	`TURE` or `FALSE`. Whether to use TSA package to perfor the diagnostics.
`...`	Additional arguments for `tsdiag` function.

Value

None. Diagnostics are plotted.

Author(s)

Ziqian Zhou

See Also

`tsdiag`

`DSIR.fit` *Fitted Delay SIR model*

Description

Fitted results of the Delay SIR model with data of the `DSIR.data` and fitted using the code in the example section of `Profile.LS.DDE`.

Usage

```
DSIR.fit
```

Format

A list returned by `Profile.LS.DDE`

`DSIRdata` *Simulated dataset from a delay SIR model*

Description

A simulated dataset from a delay SIR model specified as following:

$$\dot{S} = \rho(t) - \beta I_\tau S \dot{I} = \beta I_\tau S - \gamma I$$

The parameters are set at: $\rho(t) = 4000 \times (\sin(t) + 2)$, $\gamma = 5$, $\beta = 0.0012$, and $\tau = 0.5$. We simulated the numerical solution from time $t = 0$ to $t = 50$ and the process is then sampled regularly ten times per unit time. Independent normal observational noise with sd = 100 is added to the numerical solution.

Usage

```
DSIRdata
```

Format

A matrix of two columns. The first column is the observed state `S` and the second column is the observed state `I`.

DSIRfn.make *Make DTVSIR functions*

Description

Make DTVSIR functions

Usage

```
DSIRfn.make()
```

Value

A list of functions that calculate the derivatives of delay SIR model with time varying coefficient.

Examples

```
DSIRfn.make()
 DSIRfn <- DSIRfn.make()
```

DSIRInitCoefs *Initial values for the spline coefficients for the Delay SIR model*

Description

Initial values for the spline coefficients for the Delay SIR model used in the example of `Profile.LS.DDE` with data of the `DSIR.data`.

Usage

```
DSIRInitCoefs
```

Format

A matrix. Two columns for state "S" and "I" respectively.

DTVSIRfn.make *Make DTVSIR functions*

Description

Make DTVSIR functions

Usage

```
DTVSIRfn.make()
```

Value

A list of functions that calculate the derivatives of delay SIR model with time varying coefficient.

Examples

```
DTVSIRfn.make()
DTVSIRfn <- DTVSIRfn.make()
```

forecast.DDE *Compute and Plot the Forecasts Based on a Fitted DDE Model*

Description

The fitted process and forecasts are plotted. These may be either plotted simultaneously, as `matplot` does for multivariate data, or one by one with a mouse click to move from one plot to another. The function also accepts the other `plot` specification arguments that the regular plot does. The forecast is done by integreting the process forward and forecast confidence is based on an arima model fitted to the residual using `auto.arima` function from `forecast` package.

Usage

```
forecast.DDE(y, times, h, pars, beta, proc, more, tau, ndelay, fdobj0, fdobj.d,
  ask = FALSE, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
  axes = NULL, ...)
```

Arguments

`y`	The observed data matrix
`times`	A sequence of time points at which data are observed. must be evenly spaced to be able to fit arima model to the residual.
`h`	How many lags to predict forward.
`pars`	The fitted parameters for the DDE.
`beta`	The fitted parameters for the contribution of lags of delays.

proc	The proc object returned from estimation functions.
more	An object specifying additional arguments to fn.
tau	A list of delay lags.
ndelay	A vector inidicating which state process has a delay term.
fdobj0	A functional data object fitted with the initial history part of the data.
fdobj.d	A functional data object fitted by generalized profiling.
ask	a logical value: If TRUE, each curve is shown separately, and the plot advances with a mouse click.
xlab	a label for the horizontal axis.
ylab	a label for the vertical axis.
xlim	a vector of length 2 containing axis limits for the horizontal axis.
ylim	a vector of length 2 containing axis limits for the vertical axis.
axes	Either a logical or a list or NULL. ##'
	logical whether axes should be drawn on the plot
	list a list used to create custom axes used to create axes via do.call(x$axes[[1]], x$axes[-1]).
...	additional plotting arguments that can be used with function plot.

Value

'done'

See Also

IntegrateForward.DDE

`inneropt.DDE` *Inner optimization for estmating coefficients given parameters.*

Description

Estmates spline coefficients given parameters for DDE models.

Usage

```
inneropt.DDE(data, times, pars, beta, coefs, lik, proc, in.meth = "nlminb",
  control.in = list(), basisvals, fdobj0)
```

Arguments

data	Matrix of observed data values.
times	Vector observation times for the data.
pars	Initial values of parameters to be estimated processes.
beta	Initial values of the contribution of lags for the delay.
coefs	Vector giving the current estimate of the coefficients in the spline.
lik	lik object defining the observation process.
proc	proc object defining the state process.
in.meth	Inner optimization function currently one of 'nlminb', 'optim', or 'trustOptim'.
control.in	Control object for inner optimization function.
basisvals	Values of the collocation basis to be used. This should be a basis object from the fda package.
fdobj0	A functional data object fitted with the history part of the data.

Details

This minimizes the objective function for DDE models defined by the addition of the lik and proc objectives with respect to the coefficients. A number of generic optimization routines can be used and some experimentation is recommended.

Value

A list with elements

coefs A matrix giving he optimized coefficients.

res The results of the inner optimization function.

Author(s)

Ziqian Zhou

IntegrateForward.DDE *Intergrate Forward a DDE Model*

Description

Solves a delay differential equation going forward based on a proc object.

Usage

```
IntegrateForward.DDE(times.forecast, pars, beta, proc, more, tau, ndelay,
  fdobj0, fdobj.d)
```

Arguments

times.forecast	A time series at which the state of the process is predicted
pars	Estimated parameters.
beta	Estimated contributions of lags of delay.
proc	The proc object returned from estimation functions.
more	An object specifying additional arguments to fn.
tau	A list of delay lags.
ndelay	A vector inidicating which state process has a delay term.
fdobj0	A functional data object fitted with the initial history part of the data.
fdobj.d	A functional data object fitted by generalized profiling.

Value

A list of objects

times The time points at where the predictions are made.

states The predicted states of the process.

Author(s)

Ziqian Zhou

---

make.vector.disease.fn

Make Vector Disease functions

Description

Make Vector Disease functions

Usage

```
make.vector.disease.fn()
```

Value

A list of functions that calculate the derivatives of vector disease model.

Examples

```
make.vector.disease.fn()
vector.disease.fn <- make.vector.disease.fn()
```

`Profile.LS.DDE` *Profile Estimation Functions for DDE*

Description

This function runs generalized profiling for DDE models. This function carry out the profiled optimization method for DDe models using a sum of squared errors criteria for both fit to data and the fit of the derivatives to a delay differential equation.

Usage

```
Profile.LS.DDE(fn, data, times, pars, beta, coefs = NULL, basisvals = NULL,
  lambda, fd.obj = NULL, more = NULL, weights = NULL, quadrature = NULL,
  in.meth = "nlminb", out.meth = "nls", control.in = list(),
  control.out = list(), eps = 1e-06, active = NULL, posproc = FALSE,
  poslik = FALSE, discrete = FALSE, names = NULL, sparse = FALSE,
  basisvals0 = NULL, coefs0 = NULL, nbeta, ndelay, tau)
```

Arguments

`fn` A named list of functions giving the righthand side of a delay differential equation. The functions should have arguments

times he times at which the righthand side is being evaluated.

x The state values at those times.

p Parameters to be entered in the system.

more A list object containing additional inputs to `fn`, The distributed delay state are passed into derivative calculation as `more$y`. The list of functions should contain the elements:

fn Function to calculate the right hand sid.

dfdx Function to calculate the derivative of each right-hand function with respect to the states.

dfdp calculates the derivative of therighthand side function with respect to parameters.

d2fdx2 Function to calculate the second derivatives with respect to states.

d2fdxdp Function to calculate the cross derivatives of each right-hand function with respect to state and parameters.

dfdx.d Function to calculate the the derivative of each righthand function with respect to the delayed states.

d2fdx.ddp Function to calculate the cross derivatives of each righthand function with respect to the delayed states and parameters.

d2fdxdx.d Function to calculate the cross derivatives of each right-hand function with respect to the state and the delayed states.

d2fdx.d2 Function to calculate the second derivatives of the right-hand function with respect to the delayed states.

`data` Matrix of observed data values.

times	Vector observation times for the data.
pars	Initial values of parameters to be estimated processes.
beta	Initial values of the contribution of lags for the delay.
coefs	Vector giving the current estimate of the coefficients in the spline.
basisvals	Values of the collocation basis to be used. This should be a basis object from the fda package.
lambda	Penalty value trading off fidelity to data with fidelity to dif- ferential equations.
fd.obj	A functional data object; if this is non-null, coefs and basisvals is extracted from here.
more	An object specifying additional arguments to fn.
weights	Weights for weighted estimation.
quadrature	Quadrature points, should contain two elements (if not NULL) **qpts** sQuadrature points; defaults to midpoints between knots **qwts** Quadrature weights; defaults to normalizing by the length of qpts.
in.meth	Inner optimization function currently one of 'nlminb', 'optim', or 'trustOptim'.
out.meth	Outer optimization function to be used, depending on the type of method. **nls** Nonlinear least square **nnls.eq** Nonlinear least square with equality or/and inequality constraints of the parameters.
control.in	Control object for inner optimization function.
control.out	Control object for outer optimization function.
eps	Finite differencing step size, if needed.
active	Incides indicating which parameters of pars should be estimated; defaults to all of them.
posproc	Should the state vector be constrained to be positive? If this is the case, the state is represented by an exponentiated basis expansion in the proc object.
poslik	Should the state be exponentiated before being compared to the data? When the state is represented on the log scale (posproc=TRUE), this is an alternative to taking the log of the data.
discrete	Is it a discrete process?
names	The names of the state variables if not given by the column names of coefs.
sparse	Should sparse matrices be used for basis values? This option can save memory when using 'trust' optimization method.
basisvals0	Values of the collocation basis to be used for the history part of the data. This should be a basis object from the fda package.
coefs0	Vector giving the estimate of the coefficients in the spline for the history part of the data.
nbeta	The number of lags for the delay.
ndelay	A vector inidicating which state process has a delay term.
tau	A list of delay lags.

Value

A list with elements

data The matrix for the observed data.

res The inner optimization result.

ncoefs The estimated coefficients.

lik The `lik` object generated.

proc The `proc` object generated.

pars The estimated parameters.

beta The estimated contribution of lags for the delay.

times The times at which the data are observed.

fdobj.d The functional data object for the estimated state process.

fdobj0 The functional data object for the estimated state process of the history part.

tau The lags of delays.

Author(s)

Ziqian Zhou

Examples

```
yout <- DSIRdata
times <- seq(-0.5, 30, by = 0.1)
yout0 <- yout[times >= 0, ]
yout.d <- yout[times >= 5, ]
colnames(yout.d) <-  c("S","I")
times0 <- times[times>=0]
times.d <- times[times>=5]
norder = 3
nbasis.d = length(times.d) + norder - 2
nbasis0 <- length(times0) + norder - 2
basis0 <- create.bspline.basis(range=range(times0),
    nbasis=nbasis0, norder=norder, breaks=times0)
basis.d <- create.bspline.basis(range=range(times.d),
    nbasis=nbasis.d, norder=norder, breaks=times.d)
fdnames=list(NULL,c('S', 'I'),NULL)
bfdPar0 = fdPar(basis0,lambda=1,int2Lfd(1))
bfdPar.d <- fdPar(basis.d,lambda=1,int2Lfd(1))
DEfd0 <- smooth.basis(times0, yout0, bfdPar0,fdnames=fdnames)$fd
coefs0 <- DEfd0$coefs
colnames(coefs0) = c("S","I")
initPars <- c(5, 0.0012)
names(initPars) <- c("gamma", "beta")
initBeta <- rep(0, 11)
initBeta[c(4,5,11)] <- c(0.611, 0.362, 0.026)
tau <- list(seq(0,1, length.out = 11))
lambda = 1000
DSIRfn <- DSIRfn.make()
```

```
## Not run:
dde.fit <- Profile.LS.DDE(DSIRfn, yout.d, times.d, pars = initPars,
    beta = initBeta, coefs = DSIRInitCoefs, basisvals = basis.d,
    lambda = 1000,
    in.meth='nlminb', basisvals0 = basis0, coefs0 = coefs0,
    nbeta = length(initBeta), ndelay = 2, tau = tau,
    control.out = list(method = "nnls.eq", maxIter = 2, echo = TRUE))

## End(Not run)
```

`Profile.LS.TV.DDE` *Profile Estimation Functions for DDE with Time Varying Coefficients.*

Description

This function runs generalized profiling for DDE models with time varying coefficients. This function carry out the profiled optimization method for DDe models using a sum of squared errors criteria for both fit to data and the fit of the derivatives to a delay differential equation with time varying coefficients.

Usage

```
Profile.LS.TV.DDE(fn, data, times, pars, beta, kappa, coefs = NULL,
  basisvals = NULL, lambda, fd.obj = NULL, more = NULL, weights = NULL,
  quadrature = NULL, in.meth = "nlminb", out.meth = "nls",
  control.in = list(), control.out = list(), eps = 1e-06, active = NULL,
  posproc = FALSE, discrete = FALSE, poslik = FALSE, names = NULL,
  sparse = FALSE, basisvals0 = NULL, coefs0 = NULL, nbeta, ndelay, tau)
```

Arguments

`fn`

fn A named list of functions giving the righthand side of a delay differential equation. The functions should have arguments

times he times at which the righthand side is being evaluated.

x The state values at those times.

p Parameters to be entered in the system.

more A list object containing additional inputs to `fn`, The distributed delay state are passed into derivative calculation as `more$y`. The list of functions should contain the elements:

fn Function to calculate the right hand sid.

dfdx Function to calculate the derivative of each right-hand function with respect to the states.

dfdp calculates the derivative of therighthand side function with respect to parameters.

d2fdx2 Function to calculate the second derivatives with respect to states.

d2fdxdp Function to calculate the cross derivatives of each right-hand function with respect to state and parameters.

dfdx.d Function to calculate the the derivative of each righthand function with respect to the delayed states.

d2fdx.ddp Function to calculate the cross derivatives of each righthand function with respect to the delayed states and parameters.

d2fdxdx.d Function to calculate the cross derivatives of each right-hand function with respect to the state and the delayed states.

d2fdx.d2 Function to calculate the second derivatives of the right-hand function with respect to the delayed states.

`data` Matrix of observed data values.

`times` Vector observation times for the data.

`pars` Initial values of parameters to be estimated processes.

`beta` Initial values of the contribution of lags for the delay.

`kappa` Initial values of parameters for a time varying function.

`coefs` Vector giving the current estimate of the coefficients in the spline.

`basisvals` Values of the collocation basis to be used. This should be a basis object from the fda package.

`lambda` Penalty value trading off fidelity to data with fidelity to dif- ferential equations.

`fd.obj` A functional data object; if this is non-null, coefs and basisvals is extracted from here.

`more` An object specifying additional arguments to fn.

`weights` Weights for weighted estimation.

`quadrature` Quadrature points, should contain two elements (if not `NULL`)

qpts sQuadrature points; defaults to midpoints between knots

qwts Quadrature weights; defaults to normalizing by the length of qpts.

`in.meth` Inner optimization function currently one of `'nlminb'`, `'optim'`, or `'trustOptim'`.

`out.meth` Outer optimization function to be used, depending on the type of method.

nls Nonlinear least square

nnls.eq Nonlinear least square with equality or/and inequality constraints of the parameters.

`control.in` Control object for inner optimization function.

`control.out` Control object for outer optimization function.

`eps` Finite differencing step size, if needed.

`active` Incides indicating which parameters of pars should be estimated; defaults to all of them.

`posproc` Should the state vector be constrained to be positive? If this is the case, the state is represented by an exponentiated basis expansion in the proc object.

`discrete` Is it a discrete process.

`poslik` Should the state be exponentiated before being compared to the data? When the state is represented on the log scale (posproc=TRUE), this is an alternative to taking the log of the data.

names	The names of the state variables if not given by the column names of coefs.
sparse	Should sparse matrices be used for basis values? This option can save memory when using 'trust' optimization method.
basisvals0	Values of the collocation basis to be used for the history part of the data. This should be a basis object from the fda package.
coefs0	Vector giving the estimate of the coefficients in the spline for the history part of the data.
nbeta	The number of lags for the delay.
ndelay	A vector inidicating which state process has a delay term.
tau	A list of delay lags.

Value

A list with elements

data The matrix for the observed data.

res The inner optimization result.

ncoefs The estimated coefficients.

lik The lik object generated.

proc The proc object generated.

pars The estimated parameters.

beta The estimated contribution of lags for the delay.

kappa The estimated parameters for the time varying function.

times The times at which the data are observed.

fdobj.d The functional data object for the estimated state process.

fdobj0 The functional data object for the estimated state process of the history part.

tau The lags of delays.

Author(s)

Ziqian Zhou

ProfileSSE.covariance.DDE

ProfileSSE.covariance.DDE

Description

Newey-West estimate of covariance of parameter estimates from profiling for DDE models. Currently assumes a lag-5 auto-correlation among observation vectors.

Usage

```
ProfileSSE.covariance.DDE(pars, beta, active = NULL, eps = 1e-06, ...)
```

Arguments

pars	The estimated parameters.
beta	The estimated parameters.
active	Incides indicating which parameters of pars should be estimated; defaults to all of them.
eps	Step-size for finite difference estimate of second derivatives.
...	Additional arguments used for profiling estimation

Value

Returns a Newey-West estimate of the covariance matrix of the parameter estimates.

Author(s)

Ziqian Zhou

See Also

Profile.LS.DDE

---

sparse.DDE *Sparsity selection for the lags of delay.*

Description

This function carry out one step sparsity selection for the lags of delay given the profiled optimization result.

Usage

```
sparse.DDE(fn, data, times, basisvals = NULL, lambda, fd.obj = NULL,
  more = NULL, weights = NULL, quadrature = NULL, in.meth = "nlminb",
  out.meth = "nls", control.in = list(), control.out = list(),
  eps = 1e-06, active = NULL, posproc = FALSE, poslik = FALSE,
  names = NULL, sparse = FALSE, discrete = FALSE, basisvals0 = NULL,
  coefs0 = NULL, nbeta, ndelay, tau, nnls.res)
```

Arguments

fn	A named list of functions giving the righthand side of a delay differential equation. The functions should have arguments **times** he times at which the righthand side is being evaluated. **x** The state values at those times. **p** Parameters to be entered in the system. **more** A list object containing additional inputs to fn, The distributed delay state are passed into derivative calculation as more$y. The list of functions should contain the elements: **fn** Function to calculate the right hand sid. **dfdx** Function to calculate the derivative of each right-hand function with respect to the states. **dfdp** calculates the derivative of therighthand side function with respect to parameters. **d2fdx2** Function to calculate the second derivatives with respect to states. **d2fdxdp** Function to calculate the cross derivatives of each right-hand function with respect to state and parameters. **dfdx.d** Function to calculate the the derivative of each righthand function with respect to the delayed states. **d2fdx.ddp** Function to calculate the cross derivatives of each righthand function with respect to the delayed states and parameters. **d2fdxdx.d** Function to calculate the cross derivatives of each right-hand function with respect to the state and the delayed states. **d2fdx.d2** Function to calculate the second derivatives of the right-hand function with respect to the delayed states.
data	Matrix of observed data values.
times	Vector observation times for the data.
basisvals	Values of the collocation basis to be used. This should be a basis object from the fda package.
lambda	Penalty value trading off fidelity to data with fidelity to dif- ferential equations.
fd.obj	A functional data object; if this is non-null, coefs and basisvals is extracted from here.
more	An object specifying additional arguments to fn.
weights	Weights for weighted estimation.
quadrature	Quadrature points, should contain two elements (if not NULL) **qpts** sQuadrature points; defaults to midpoints between knots **qwts** Quadrature weights; defaults to normalizing by the length of qpts.
in.meth	Inner optimization function currently one of 'nlminb', 'optim', or 'trustOptim'.
out.meth	Outer optimization selection function to be used, depending on the type of method. **"penalized"** Uses LASSO method from penalized package. **"addaptive"** Positive addaptive lasso using lars algorithm.

"lars" Positive lasso using lars algorithm.

`control.in` Control object for inner optimization function.

`control.out` Control object for outer optimization function.

`eps` Finite differencing step size, if needed.

`active` Incides indicating which parameters of pars should be estimated; defaults to all of them.

`posproc` Should the state vector be constrained to be positive? If this is the case, the state is represented by an exponentiated basis expansion in the proc object.

`poslik` Should the state be exponentiated before being compared to the data? When the state is represented on the log scale (posproc=TRUE), this is an alternative to taking the log of the data.

`names` The names of the state variables if not given by the column names of coefs.

`sparse` Should sparse matrices be used for basis values? This option can save memory when using 'trust' optimization method.

`discrete` Is it a discrete process?

`basisvals0` Values of the collocation basis to be used for the history part of the data. This should be a basis object from the fda package.

`coefs0` Vector giving the estimate of the coefficients in the spline for the history part of the data.

`nbeta` The number of lags for the delay.

`ndelay` A vector inidicating which state process has a delay term.

`tau` A list of delay lags.

`nnls.res` `res` item returned from `Profile.LS.DDE`

Value

A list with elements

data The matrix for the observed data.

res The inner optimization result.

select A list containing the result after selection, the parameter, delay contribution and coefficients after the selection.

Author(s)

Ziqian Zhou

See Also

`Profile.LS.DDE`

sparse.TV.DDE *Sparsity selection for the lags of delay and time varying coefficients*

Description

Sparsity selection for the lags of delay and time varying coefficients This function carry out one step sparsity selection for the lags of delay given the profiled optimization result.

Usage

```
sparse.TV.DDE(fn, data, times, basisvals = NULL, lambda, fd.obj = NULL,
  more = NULL, weights = NULL, quadrature = NULL, in.meth = "nlminb",
  out.meth = "nls", control.in = list(), control.out = list(),
  eps = 1e-06, active = NULL, posproc = FALSE, poslik = FALSE,
  discrete = FALSE, names = NULL, sparse = FALSE, basisvals0 = NULL,
  coefs0 = NULL, nbeta, ndelay, tau, nnls.res)
```

Arguments

fn — A named list of functions giving the righthand side of a delay differential equation. The functions should have arguments

times he times at which the righthand side is being evaluated.

x The state values at those times.

p Parameters to be entered in the system.

more A list object containing additional inputs to fn, The distributed delay state are passed into derivative calculation as more$y. The list of functions should contain the elements:

fn Function to calculate the right hand sid.

dfdx Function to calculate the derivative of each right-hand function with respect to the states.

dfdp calculates the derivative of therighthand side function with respect to parameters.

d2fdx2 Function to calculate the second derivatives with respect to states.

d2fdxdp Function to calculate the cross derivatives of each right-hand function with respect to state and parameters.

dfdx.d Function to calculate the the derivative of each righthand function with respect to the delayed states.

d2fdx.ddp Function to calculate the cross derivatives of each righthand function with respect to the delayed states and parameters.

d2fdxdx.d Function to calculate the cross derivatives of each right-hand function with respect to the state and the delayed states.

d2fdx.d2 Function to calculate the second derivatives of the right-hand function with respect to the delayed states.

data — Matrix of observed data values.

times	Vector observation times for the data.
basisvals	Values of the collocation basis to be used. This should be a basis object from the fda package.
lambda	Penalty value trading off fidelity to data with fidelity to dif- ferential equations.
fd.obj	A functional data object; if this is non-null, coefs and basisvals is extracted from here.
more	An object specifying additional arguments to fn.
weights	Weights for weighted estimation.
quadrature	Quadrature points, should contain two elements (if not NULL) **qpts** sQuadrature points; defaults to midpoints between knots **qwts** Quadrature weights; defaults to normalizing by the length of qpts.
in.meth	Inner optimization function currently one of 'nlminb', 'optim', or 'trustOptim'.
out.meth	Outer optimization selection function to be used, depending on the type of method. **"penalized"** Uses LASSO method from penalized package.
control.in	Control object for inner optimization function.
control.out	Control object for outer optimization function.
eps	Finite differencing step size, if needed.
active	Incides indicating which parameters of pars should be estimated; defaults to all of them.
posproc	Should the state vector be constrained to be positive? If this is the case, the state is represented by an exponentiated basis expansion in the proc object.
poslik	Should the state be exponentiated before being compared to the data? When the state is represented on the log scale (posproc=TRUE), this is an alternative to taking the log of the data.
discrete	Is it a discrete process.
names	The names of the state variables if not given by the column names of coefs.
sparse	Should sparse matrices be used for basis values? This option can save memory when using 'trust' optimization method.
basisvals0	Values of the collocation basis to be used for the history part of the data. This should be a basis object from the fda package.
coefs0	Vector giving the estimate of the coefficients in the spline for the history part of the data.
nbeta	The number of lags for the delay.
ndelay	A vector inidicating which state process has a delay term.
tau	A list of delay lags.
nnls.res	nnls.res res item returned from Profile.LS.DDE

Value

A list with elements

data The matrix for the observed data.

res The inner optimization result.

select A list containing the result after selection, the parameter, delay contribution and coefficients after the selection.

Author(s)

Ziqian Zhou

See Also

`Profile.LS.TV.DDE`

Index