Package 'odest'

March 4, 2015

Type Package

Title Reproducing kernel Hilbert Space based estimation of parameters of systems of Ordinary Differential equations

Version 1.0

Date 2013-05-14

Author Ivan Vujacic <i.vujacic@rug.nl>, Javier Gonzalez <j.gonzalez.hernandez@rug.nl>, Ernst Wit <e.c.witg@rug.nl>

Depends R (>= 2.10.1), expm, pspline, magic, MASS, corpcor, gplots,deSolve

Maintainer Ivan Vujacic <i.vujacic@rug.nl>

Description These functions implement RKHS based estimation of parameters of ODEs. They provide parameter estimates, confidence intervals and estimates of state variables.

LazyLoad yes

R topics documented:

odest-package 2
ci.boot 2
Exponentialdata 3
FhNdata 3
lambda.select 4
LVdata 4
plot.odesol 5
rkhs 5

Index **11**

odest-package — *Reproducing Kernel Hilbert Space based estimation of Systems of Ordinary Differential Equations*

Description

These functions implement RKHS based estimation for Systems of Ordinary Differential Equations. They provide estimate of parameters, confidence intervals and estimates of state variables.

Details

Package:	oderkhs
Type:	Package
Version:	1.0
Date:	2013-05-14
License:	GPL (>= 3)
LazyLoad:	yes

Author(s)

Ivan Vujacic, Javier Gonzalez, Ernst Wit

Maintainer: Ivan Vujacic <i.vujacic@rug.nl> and Javier Gonz<c3><a1>lez <j.h.gonzalez@sheffield.ac.uk>

ci.boot — *Estimating Confidence Intervals*

Description

Estimating Confidence Intervals

Usage

```
ci.boot(rkhs.obj, nboot = 10)
```

Arguments

`rkhs.obj`	List returned by rkhs function.
`nboot`	Number of bootstrap samples.

Value

A list with the following components:

`data`	Original data.
`ci`	Confidence intervals of parameters.
`par`	Estimated parameters obtained by function 'rkhs'.
`par.boot`	Bootstrapped parameters.
`times`	Vector of observation times for the data.
`x.hat`	Estimate of state variables obtained by function 'rkhs'.
`object`	Set to 'ci' and indicates that the list is provided by function 'ci.boot'.
`varnames`	Vector that contains the names of state variables.
`ODEmod`	A function that defines ODE model.

`Exponentialdata` *Data generated from linear differential equation of first order*

Usage

```
data(Exponentialdata)
```

Format

- Exponentialdata A 5 by 2 matrix of data generated from linear differential equation of first order with true value equal to theta=-2. First column contains vector of observation times and second and third contain observations of state variables.

Source

Javier Gonzalez, Ivan Vujacic and Ernst Wit, 2012. "A new statistical framework to infer gene regulatory networks with hidden transcription factors". Revised and Resubmitted.

`FhNdata` *Data generated from FitzHugh-Nagumo equations*

Usage

```
data(FhNdata)
```

Format

- FhNdata A 50 by 3 matrix of data generated from the FitzHugh Nagumo equations with true parameter theta = (0.2,0.2,3). First column contains vector of observation times and second and third contain observations of state variables.

Source

James Ramsay, Giles Hooker David Campbell and Jiguo Cao, 2007. "Parameter Estimation for Differential Equations: A Generalized Smoothing Approach". Journal of the Royal Statistical Society Vol 69 No 5.

`lambda.select` *Selecting the regularization parameter with AIC or GCV*

Description

Selection of the regularization parameter

Usage

```
lambda.select(rkhs.obj, lambda.grid=seq(1,10^4,length=10),
criterion = "AIC", optimise = TRUE)
```

Arguments

`rkhs.obj`	List returned by the function rkhs.
`lambda.grid`	Grid of lambda values in which selection criterion is to be evaluated if optimise=FALSE.
`criterion`	Selection criterion. Defult value is "AIC" (Akaike's Information Criterion) and the other option is "GCV" (Generalized Cross-Validation).
`optimise`	Logical value that indicates if selection criterion should be optimised over all positive real numbers.

Value

Returns optimal value of lambda in terms of the criterion that is selected.

`LVdata` *Data generated from Lotka Volterra equations*

Usage

```
data(LVdata)
```

Format

- LVdata A 30 by 3 matrix of data generated from the Lotka Volterra equations with true parameter theta = (0.2,0.35,0.7,0.40). First column contains vector of observation times and second and third contain observations of state variables.

`plot.odesol` *Plots state variables and confidence bands of state variables. Confidence bands are based on the variance estimated from the residuals. Note: Confidence bands can be very wide.*

Description

Plots estimated state variables if the object is returned by 'rkhs' function and confidence bands of state variables if the object is returnd by 'ci.boot' function.

Usage

```
plot.odesol(object)
```

Arguments

`object`	List either returned by the function 'rkhs' or the function 'ci.boot'.

`rkhs` *rkhs.estimation*

Description

RKHS estimation of ODE system

Usage

```
rkhs(data, f, coef, times, lambda, par, sigma = NULL,varnames,ODEmod)
```

Arguments

`data`	Matrix of observed data values.
`f`	List giving the right hand side of each equation.
`coef`	List giving the coefficients of differential operators for each equation.
`times`	Vector of observation times for the data.
`lambda`	Regularization parameter.
`par`	Initial values of parameters to be estimated.
`sigma`	If vector, it comprises variances of each state variable. If scalar, then variance of each state variable is assumed to be equal to given value. If NULL, it is estimated offline.
`varnames`	Vector that contains the names of state variables.
`ODEmod`	A function that defines ODE model.

Value

A list with where the most important component is:

`par`	Estimated parameters.
`data`	Matrix of observed data values.
`f`	List giving the right hand side of each equation.
`coef`	List giving the coefficients of differential operators for each equation.
`times`	Estimated parameters.
`x.hat`	Estimated state variables.
`lambda`	Regularization parameter.
`Sigma`	Diagonal matrix of variances for every state.
`Slambda`	Smoother matrix.
`object`	Set to 'rkhs' and indicates that the list is provided by function 'rkhs'.
`varnames`	Vector that contains the names of state variables.
`ODEmod`	A function that defines ODE model.

Source

Javier Gonzalez, Ivan Vujacic and Ernst Wit. Reproducing kernel Hilbert space based estimation of systems of ordinary differential equations. Pattern Recognition Letters, 45(1), 26–32, 2014.

Javier Gonzalez, Ivan Vujacic and Ernst Wit. Inferring latent gene regulatory network kinetics. Statistical Applications in Genetics and Molecular Biology. Vol. 12, Issue 1, pp. 109–127, 2013.

Examples

```
###########################
##FitzHugh-Nagumo example##
###########################
#setting up the system
f1 = function( x, theta )
{
f = matrix(-x[,1]^3/3+x[,2],ncol=1)
return(f)
}
f2 = function( x, theta )
{
f = matrix( -x[,1]+rep( theta[1],length(x[,1]) ) ,ncol=1)
return(f)
}
coef1 = function(theta)
{
return( c(-1,1/theta[3]) )
}
coef2 = function(theta)
{
return( c(theta[2],theta[3]) )
```

```
}
f=list(f1,f2)
coef = list(coef1,coef2)
#ODE
FHNmod <- function(Time, State, Pars) {
with(as.list(c(State, Pars)), {
dx <- c*(x-x^3/3 + y)
dy <- -1/c*(x-a+b*y)
return(list(c(dx, dy)))
})
}
#true parameters and initial conditions
pms = c(a = 0.2, b = 0.2, c = 3)
yini = c(x = -1, y = 1)
#sample size
n = 50
#level of noise
noise = 0.1
times = seq(0, 30, length = n ) ## sequence of t1,...,tn
out = ode(yini, times, FHNmod, pms) ## ODE solution
## Generate the data
x = out[,2:3]
y = as.matrix(cbind(out[,2] + rnorm(n,0,noise),out[,3] +rnorm(n,0,noise)))
#variable names and initial parameters
par=c(a=4, b=4, c=4)
varnames=c(x,y)
colnames(y)=varnames
# estimate with lambda=1
res=rkhs(data=y,f=f,coef=coef,times=times,lambda=1,par=par,varnames=varnames,ODEmod=FHNmod)
res$par
#selection of lambda with AIC
lambda=lambda.select(res,optimise=FALSE)
##or slower option
#lambda=lambda.select(res)
res=rkhs(data=y,f=f,coef=coef,times=times,lambda=lambda,par=par,
varnames=varnames,ODEmod=FHNmod)
res$par
plot.odesol(res)
res.boot = ci.boot(res,20)
res.boot$ci
plot.odesol(res.boot)


################################
##Simple linear system example##
################################
#setting up the system
coef1 = function(theta)
{
return(c(-theta,1) )
}
f1 = function(x, theta)
{
```

```
f = 0*x
return(f)
}
f=list(f1)
coef = list(coef1)
separable=NULL
#ODE
EXPmod <- function(Time, State, Pars) {
with(as.list(c(State, Pars)), {
dx <- theta * x
return(list(c(dx)))
})
}
#true parameters and initial conditions
pms = c(theta = -2)
yini = c(x = -1)
#sample size
n = 50
#level of noise
noise = 0.1
times = seq(0, 1, length = n ) ## sequence of t1,...,tm
out = ode(yini, times, EXPmod, pms) ## ODE solution
## Generate the data
x = out[,2]
y = as.matrix(cbind(out[,2] + rnorm(n,0,noise)))
#variable names and initial parameters
par=c(theta=1)
varnames=c(x)
colnames(y)=varnames
# estimate with lambda=1000
res=rkhs(data=y,f=f,coef=coef,times=times,lambda=1000,par=par,
varnames=varnames,ODEmod=EXPmod)
res$par
#selection of lambda with AIC
lambda=lambda.select(res,optimise=FALSE)
##or slower option
#lambda=lambda.select(res)
res=rkhs(data=y,f=f,coef=coef,times=times,lambda=lambda,par=par,
varnames=varnames,ODEmod=EXPmod)
res$par
# confidence intervals
res.boot = ci.boot(res,20)
res.boot$ci
# plot estimate of state variable
plot.odesol(res)
#plot confidence bands
plot.odesol(res.boot)

##########################
##Lotka-Volterra example##
##########################
#setting up the system
f1 = function( x, theta )
```

```
{
f = -theta[2]*matrix(x[,1]*x[,2],ncol=1)
return(f)
}
f2 = function( x, theta )
{
f = theta[4]*matrix(x[,1]*x[,2],ncol=1)
return(f)
}
System.f=list(f1,f2)
coef1 = function(theta)
{
return( c(-theta[1],1) )
}
coef2 = function(theta)
{
return( c(theta[3],1) )
}
f = list(f1,f2)
coef = list(coef1,coef2)
separable=NULL
#ODE
LVmod <- function(Time, State, Pars) {
with(as.list(c(State, Pars)), {
dx <- alpha * x - beta *x*y
dy <- -gamma*y + delta*x*y
return(list(c(dx, dy)))
})
}
#true parameters and initial conditions
pms = c(alpha = 0.2,beta = 0.35, gamma = 0.7, delta = 0.40 )
yini = c(x = 1, y = 2)
## sample size values
n = 50
## level of noise
noise = 0.1
times = seq(0, 30, length = n ) ## sequence of t1,...,tm
out = ode(yini, times, LVmod, pms) ## ODE solution
## Generate the data
x = out[,2:3]
y = as.matrix(cbind(out[,2] + rnorm(n,0,noise),out[,3] +rnorm(n,0,noise)))
#variable names and initial parameters
par=c(alpha = 4,beta = 4, gamma = 4, delta = 4 )
varnames=c(x,y)
colnames(y)=varnames
#estimate with lambda=1
res=rkhs(data=y,f=f,coef=coef,times=times,lambda=1,par=par,
varnames=varnames,ODEmod=LVmod)
res$par
#selection of lambda with AIC
lambda=lambda.select(res,optimise=FALSE)
##or slower option
#lambda=lambda.select(res)
```

```
res=rkhs(data=y,f=f,coef=coef,times=times,lambda=lambda,par=par,
varnames=varnames,ODEmod=LVmod)
res$par
# confidence intervals
res.boot = ci.boot(res,20)
res.boot$ci
#plot estimates of state variables
plot.odesol(res)
#plot confidence bands
plot.odesol(res.boot)
```

Index

*Topic **datasets**
    Exponentialdata, 3
    FhNdata, 3
    LVdata, 4

ci.boot, 2

Exponentialdata, 3

FhNdata, 3

lambda.select, 4
LVdata, 4
LVNdata *(LVdata)*, 4

odest *(odest-package)*, 2
odest-package, 2

plot.odesol, 5

rkhs, 5