Package: longitudinalANAL (via r-universe)

November 6, 2024

Type Package

Title Longitudinal Data Analysis

Version 0.2

Description Regression analysis of mixed sparse synchronous and asynchronous longitudinal covariates. Please cite the manuscripts corresponding to this package: Sun, Z. et al. (2023) <arXiv:2305.17715> and Liu, C. et al. (2023) <arXiv:2305.17662>.

License GPL-3

Encoding UTF-8

Depends dplyr, tibble, MASS, dlm

RoxygenNote 7.2.3

NeedsCompilation no

Author Zhuowei Sun [aut, cre, cph], Congmin Liu [aut], Chen Li [aut], Hongyuan Cao [aut, ctb], Jason Peter Fine [aut]

Maintainer Zhuowei Sun <sunzw21@mails.jlu.edu.cn>

Repository CRAN

Date/Publication 2023-10-13 12:00:02 UTC

Contents

lda *Longitudinal data analysis*

Description

This function provide regression analysis of mixed sparse synchronous and asynchronous longitudinal covariates.

Usage

```
lda(data_res, data_cov, N, bd, omit, method)
```

Arguments

data_res	An object of class tibble. The structure of the tibble must be: tibble(id_y=ID, ty=measurement time for response, y=observation for response, x=matrix(observation for synchronous covariates), x_add=matrix(observation for uninterested synchronous covariates)).
data_cov	An object of class tibble. The structure of the tibble must be: tibble(id_z=ID, tz=measurement time for response, z=matrix(observation for asynchronous covariates)).
N	An object of class integer. The sample size.
bd	An object of class vector. If use auto bandwidth selection, the structure of the vector must be: d=c(the maximum bandwidth, the minimum bandwidth, the fold of cross-validation, the number of bandwidth divided). If use fixed bandwidth, bd=c(the chosen bandwidth).
omit	An object of class integer indicating the method used to do estimation for synchronous covariates. If use plm method, omit=1; if use centering method, omit=2; if use additional covariates information, omit=3.
method	An object of class integer indicating the method used to do estimation for asynchronous covariates. If only deal with omit variable, method=0; if use two-stage method, method=1; if use kernel smoothing, method=2.

Value

a list with the following elements:

est	The estimation for the corresponding parameters.
se	The estimation of standard error for the estimated parameters.

Examples

```
library(MASS)
library(tibble)
library(dplyr)
N=100
```

```
ty=tz=y=x=z=id_y=id_z=list()
a=b=g=1
ny=rpois(N,5)+1
nz=rpois(N,5)+1
for(i in 1:N){
  ty[[i]]=as.matrix(runif(ny[i]))
  tz[[i]]=as.matrix(runif(nz[i]))
  t.temp=rbind(tz[[i]],ty[[i]])
  n.temp=nz[i]+ny[i]
  corr=exp(-abs(rep(1,n.temp)%*%t(t.temp)-t.temp%*%t(rep(1,n.temp))))
  corr.e=2^(-abs(rep(1,n.temp)%*%t(t.temp)-t.temp%*%t(rep(1,n.temp))))
  MX=t.temp^.5
  MZ=rep(0, n.temp)
  x.temp=mvrnorm(1,MX,corr)
  z.temp=mvrnorm(1,MZ, corr)
  z[[i]]=as.matrix(z.temp[1:nz[i]])
  x[[i]]=as.matrix(x.temp[-(1:nz[i])])
  id_z[[i]]=rep(i,nz[i])
  id_y[[i]]=rep(i,ny[i])
  y.temp=a+g*z.temp+x.temp*b+as.matrix(mvrnorm(1,rep(0,n.temp),corr.e))
  y[[i]]=as.matrix(y.temp[-(1:nz[i])])
}
data_cov=tibble(id_z=unlist(id_z),tz=unlist(tz),z=matrix(unlist(z),length(unlist(z))))
data_res=tibble(id_y=unlist(id_y),ty=unlist(ty),x=matrix(unlist(x),length(unlist(x))),y=unlist(y))
bd=0.1
omit=1
method=1
lda(data_res,data_cov,N,bd,omit,method)
```

`ldatv` *Longitudinal data analysis*

Description

This function provide regression analysis of mixed sparse synchronous and asynchronous longitudinal covariates with time-varying coefficients.

Usage

```
ldatv(data_res, data_cov, time, N, bd, method, scb)
```

Arguments

`data_res`	An object of class tibble. The structure of the tibble must be: tibble(id_y=ID, ty=measurement time for response, y=observation for response, x=matrix(observation for synchronous covariates), x_add=matrix(observation for uninterested synchronous covariates)).
`data_cov`	An object of class tibble. The structure of the tibble must be: tibble(id_z=ID, tz=measurement time for response, z=matrix(observation for asynchronous covariates)).

time	An object of class vector. The interest time.
N	An object of class integer. The sample size.
bd	An object of class vector. If use auto bandwidth selection, the structure of the vector must be: bd=c(the maximum bandwidth for h1, the minimum bandwidth for h1, the maximum bandwidth for h2, the minimum bandwidth for h2, the fold of cross-validation, the number of bandwidth divided). If use fixed bandwidth, bd=c(the chosen bandwidth).
method	An object of class integer indicating the method used to do estimation for asynchronous covariates. If use one-stage method, method=1; if use two-stage method with centering method for the first stage, method=1; if use two-stage method with time-varying method for the first stage, method=2.
scb	An object of class vector. If need to construct the simultaneous confidence band, the structure of the vector must be: c(alpha=desirable confidence level, B=repeat times). Otherwise, scb=0.

Value

a list with the following elements:

est.b	The estimation for the parameter of synchronous covariates.
est.g	The estimation for the parameter of asynchronous covariates.
se.b	The estimation of standard error for the parameter of synchronous covariates.
se.g	The estimation of standard error for the parameter of asynchronous covariates.
c_alpha_x	The empirical percentile used to construct the simultaneous confidence band for the parameter of synchronous covariates.
c_alpha_z	The empirical percentile used to construct the simultaneous confidence band for the parameter of asynchronous covariates.

Examples

```
library(dplyr)
library(MASS)
library(tibble)
N=400
ty=tz=y=x=x1=z=id_y=id_z=list()
beta<-function(t){
 0.3*(t-0.4)^2
}
gamma<-function(t){
 sin(2*pi*t)
}
ny=rpois(N,5)+1
nz=rpois(N,5)+1
for(i in 1:N){
 ty[[i]]=as.matrix(runif(ny[i]))
 tz[[i]]=as.matrix(runif(nz[i]))
 t.temp=rbind(tz[[i]],ty[[i]])
 n.temp=nz[i]+ny[i]
```

```
 corr=exp(-abs(rep(1,n.temp)%*%t(t.temp)-t.temp%*%t(rep(1,n.temp))))
 corr.e=2^(-abs(rep(1,n.temp)%*%t(t.temp)-t.temp%*%t(rep(1,n.temp))))
 MX=rep(0, n.temp)
 MZ= 2*(t.temp-0.5)^2
 x.temp=mvrnorm(1,MX,corr)
 z.temp=mvrnorm(1,MZ, corr)
 z[[i]]=as.matrix(z.temp[1:nz[i]])
 x[[i]]=as.matrix(x.temp[-(1:nz[i])])
 id_z[[i]]=rep(i,nz[i])
 id_y[[i]]=rep(i,ny[i])
y.temp=gamma(t.temp)*z.temp+beta(t.temp)*x.temp+as.matrix(mvrnorm(1,rep(0,n.temp),corr.e))
 y[[i]]=as.matrix(y.temp[-(1:nz[i])])
}
data_cov=tibble(id_z=unlist(id_z),tz=unlist(tz),z=matrix(unlist(z),length(unlist(z))))
data_res=tibble(id_y=unlist(id_y),ty=unlist(ty),x=matrix( unlist(x),length(unlist(x))), y=unlist(y))
ldatv(data_res,data_cov,time=0.3,N,bd=c(N^(-0.5),N^(-0.5)),method=1,scb=0)
```

Index