Package ‘nets’

October 28, 2017

Type Package
Title Network Estimation for Time Series
Version 0.9
Date 2017-10-23
Author Christian Brownlees
Maintainer Christian Brownlees <christian.brownlees@upf.edu>
Imports stats, igraph
Description Sparse VAR estimation based on LASSO.
License GPL
LazyLoad yes
URL https://github.com/ctbrownlees/R-Package-nets
Repository CRAN
NeedsCompilation yes
Date/Publication 2017-10-28 17:04:37 UTC

R topics documented:

- nets-package
- nets

Index

nets-package Network Estimator for Time Series

Description

The NETS package provides routines for the estimation of sparse VAR models.

Details
Package: nets
Type: Package
Version: 0.8
Date: 2016-03-01
License: GPL
LazyLoad: yes

Author(s)
Christian Brownlees
Maintainer: Christian Brownlees

References
Regression Model, JASA, 104, 735-746.
Meinshausen, N. and Buhlmann, P. (2006), High Dimensional Graphs and Variable Selection with

nets

Network Estimation for Time Series

Description
'nets' is used to fit sparse VARs using the NETS algorithm.

Usage
nets(y,GN=TRUE,CN=TRUE,p=1,lambda,alpha.init=NULL,rho.init=NULL,
algorithm='activeshooting',weights='adaptive',
iter.in=100,iter.out=2,verbose=FALSE)

Arguments
y data, an T x N matrix, each column being a time series.
GN Estimate the Autoregressive VAR matrices (default true)
CN Estimate the Concentration matrix of the VAR innovations (default true)
p VAR order (default 1)
lambda shrinkage parameter either a scalar or a vector of size two
alpha.init initial value of the alpha parameter
nets

rho.init  initial value of the rho parameter
algorithm  lasso optimization algorithm 'shooting' or 'activeshooting' (default)
weights  lasso weights: 'none' or 'adaptive' (default)
iter.in  maximum number of in iterations (default 100)
iter.out  maximum number of out iterations (default 2)
verbose  extra output messages

Details

The nets procedure estimates sparse Vector Autoregression (VAR) models by LASSO.
In particular, the routine can be used to estimate: (i) the autoregressive matrices of the VAR
model and the concentration matrix of the VAR innovations via the nets algorithm (GN=TRUE
and CN=TRUE) (ii) the autoregressive matrices of the VAR model via the LASSO algorithm
(GN=TRUE and CN=FALSE), (iii) the concentration matrix of data via the space algorithm of
Peng et al. (2009) (GN=FALSE and CN=TRUE). Notice that in this last case the order p of the
VAR is automatically set to zero.

In case (i) the penalty parameter lambda can be either set to a single scalar or two a vector of
size two. In the former case all the parameters of the model are penalized using the single value of
lambda provided while in the latter the first entry is the vector is used to penalize the autoregressive
coefficients and the second entry for the contemporaneous correlation coefficients. In case (ii) and
(iii) the penalty parameter lambda has to be a single scalar.

Two variants of the LASSO algorithm are implemented: "shooting" and "activeshooting". The
second can be significantly faster when the model is sparse.

If the weights variable is set to 'none' the model is estimated using the standard lasso. If the weights
variable is set to 'adaptive' the model is estimated using the 'adaptive' lasso.

The iter.in variable sets the maximum number of lasso iterations. The iter.out variable sets the
number of times the lasso algorithm is reiterated. This is only relevant for the space and nets
algorithms. See Peng et al. (2009) and Barigozzi and Brownlees (2016) for details.

The nets procedure returns an object of class "nets"

The function "print" is used to print a summary of the estimation results of a "nets" object.
The function "predict" is used to predict the future realizations of the VAR using a "nets" object and
a sample of new observation.

Value

An object of class "nets" is a list containing at least the following components:
A.hat: an N x N x P array containing the estimated autoregressive matrices
C.hat: an N x N matrix containing the estimated concentration matrix
alpha.hat: (N^2 x P) x 1 vector of autoregressive parameters stacked in a vector
rho.hat: (N(N+1)/2) x 1 vector of partial correlations associated with the concentration matrix of
the var innovations stacked in a vector
c.hat: N x 1 vector of diagonal entries of the diagonal entries of the concentration matrix of the var
innovations
g.adj: Adjacency matrix associated with the Granger network implied by the VAR autoregressive matrices

c.adj: Adjacency matrix associated with the contemporaneous network implied by the VAR innovation concentration matrix

Author(s)

Christian Brownlees

References


Examples

```r
N <- 5
P <- 3
T <- 1000

A <- array(0,dim=c(N,N,P))
C <- matrix(0,N,N)

A[,1] <- 0.7 * diag(N)
A[,2] <- 0.2 * diag(N)
A[1,2,1] <- 0.2
A[4,3,2] <- 0.2

C <- diag(N)
C[1,1] <- 2
C[4,2] <- -0.2
C[2,4] <- -0.2
C[1,3] <- -0.1
C[1,3] <- -0.1

Sig <- solve(C)
L <- t(chol(Sig))

y <- matrix(0,T,N)
eps <- rep(0,N)

for( t in (P+1):T ){
  z <- rnorm(N)
  for( i in 1:N ){
    eps[i] <- sum( L[i,] * z )
  }
}
```
for( l in 1:P ){
  for( i in 1:N ){
    y[t,i] <- y[t,i] + sum(A[i,,l] * y[t-l,])
  }
}
y[t,] <- y[t,] + eps
}

lambda <- c(1,2)

 contractual.time( mdl <- nets(y,P,lambda=lambda*T,verbose=TRUE) )

mdl
Index

*Topic multivariate timeseries
  nets, 2
*Topic network
  nets, 2
*Topic package
  nets-package, 1

nets, 2
nets-package, 1

predict.nets(nets), 2
print.nets(nets), 2