Package ‘multivar’

January 5, 2022

Encoding UTF-8
Title Penalized Estimation and Forecasting of Multiple Subject Vector Autoregressive (multi-V AR) Models
Version 1.0.0
Depends R (>= 2.10)
Imports methods, stats, utils, MASS, Rcpp (>= 1.0.3), Matrix, ggplot2, vars, reshape2
License GPL (>= 2)
LazyData true
ByteCompile true
RoxygenNote 7.1.1
NeedsCompilation yes
Maintainer Zachary Fisher <fish.zachary@gmail.com>
Repository CRAN
LinkingTo Rcpp,RcppArmadillo
VignetteBuilder knitr,rmarkdown
Suggests knitr,rmarkdown
Author Zachary Fisher [aut, cre], Younghoon Kim [ctb], Vladas Pipiras [ctb]
Date/Publication 2022-01-05 03:50:05 UTC

R topics documented:

multivar-package ................................................................. 2
canonical.multivar .............................................................. 2
constructModel ................................................................. 3
multivar-package

Penalized Estimation and Forecasting of Multiple Subject Vector Autoregressive (multi-VAR) Models

Description

multivar is an R package for simulating, estimating and forecasting stationary Vector Autoregressive (VAR) models for multiple subject data using the penalized multi-VAR framework.

Author(s)

Maintainer: Zachary Fisher <fish.zachary@gmail.com>

Other contributors:

• Younghoon Kim [contributor]
• Vladas Pipiras [contributor]

canonical.multivar

Canonical VAR Fitting Function for multivar

Description

Canonical VAR Fitting Function for multivar

Usage

canonical.multivar(object)

Arguments

object multivar object built using ConstructModel.

Details

A function to fit a canonical VAR model to each individual dataset.
Value

A list of results.

See Also

constructModel

Examples

# example 1 (run)
sim1 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)
model1 <- constructModel(data = sim1$data, weightest = "ols")
fit1 <- canonical.multivar(model1)
constructModel

standardize = T,
weightest = "ols",
canonical = FALSE,
threshold = FALSE,
lassotype = "adaptive",
intercept = FALSE,
W = NULL,
ratios = NULL
)

Arguments

data List. A list (length = k) of T by d multivariate time series
lag Numeric. The VAR order. Default is 1.
horizon Numeric. Desired forecast horizon. Default is 1. ZF Note: Should probably be zero.
t1 Numeric. Index of time series in which to start cross validation. If NULL, default is floor(nrow(n)/3) where nk is the time series length for individual k.
t2 Numeric. Index of times series in which to end cross validation. If NULL, default is floor(2*nrow(n)/3) where nk is the time series length for individual k.
lambda1 Matrix. Regularization parameter 1. Default is NULL.
lambda2 Matrix. Regularization parameter 2. Default is NULL.
nlambda1 Numeric. Number of lambda1 values to search over. Default is 30.
nlambda2 Numeric. Number of lambda2 values to search over. Default is 30.
depth Numeric. Depth of grid construction. Default is 1000.
tol Numeric. Optimization tolerance (default 1e-4).
window Numeric. Size of rolling window.
standardize Logical. Default is true. Whether to standardize the individual data.
weightest Character. Default is "mlr" for multiple linear regression. "sls" for simple linear regression also available. How to estimate the first-stage weights.
canonical Logical. Default is false. If true, individual datasets are fit to a VAR(1) model.
threshold Logical. Default is false. If true, and canonical is true, individual transition matrices are thresholded based on significance.
lassotype Character. Default is "adaptive". Choices are "standard" or "adaptive" lasso.
intercept Logical. Default is FALSE.
W Matrix. Default is NULL.
ratios Numeric vector. Default is NULL.

Examples

sim <- multivar_sim(
  k = 2, # individuals
library(multivar)

# example 1 (run)
sim1 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 2 (run)
sim2 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 3 (run)
sim3 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 4 (run)
sim4 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 5 (run)
sim5 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 6 (run)
sim6 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 7 (run)
sim7 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 8 (run)
sim8 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 9 (run)
sim9 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 10 (run)
sim10 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 11 (run)
sim11 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 12 (run)
sim12 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 13 (run)
sim13 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 14 (run)
sim14 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")

# example 15 (run)
sim15 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

plot_sim(sim, plot_type = "common")

model <- constructModel(data = sim$data, weightest = "ols")
`sigma = diag(1,3) # noise
```

```r
model1 <- constructModel(data = sim1$data, weightest = "ols")
fit1 <- multivar::cv.multivar(model1)
```

```r
## Not run:

# example 2 (don't run)
sim2 <- multivar_sim(
    k = 10, # individuals
d = 10, # number of variables
    n = 100, # number of timepoints
    prop_fill_com = 0.1, # proportion of paths common
    prop_fill_ind = 0.1, # proportion of paths unique
    lb = 0.1, # lower bound on coefficient magnitude
    ub = 0.9, # upper bound on coefficient magnitude
    sigma = diag(1,10) # noise
)

model2 <- constructModel(data = sim2$data, weightest = "ols")
fit2 <- cv.multivar(model2)
```

```r
## End(Not run)
```

---

**dat_multivar_sim**

**Simulated multi-VAR data.**

**Description**

This dataset contains multivariate time series data for $k = 9$ individuals with $d = 10$ variables collected at $t = 100$ equidistant time points. The data was generated such that each individual’s VAR(1) transition matrix has 20 percent nonzero entries. This means, for example, each individual has 20 nonzero directed relationships in their data generating model. The position of non-zero elements in each individual’s transition matrix was selected randomly given the following constraints: 2/3 of each individual’s paths are shared by all individuals, and 1/3 are unique to each individual. For each individual, coefficient values between $U(0, 0.9)$ were randomly drawn until stability conditions for the VAR model were satisfied.

**Usage**

`dat_multivar_sim`

**Format**

A list containing

- `mat_com` a common effects transition matrix
**multivar-class**

- **mat_ind_unique** a list of unique (individual-specific) effect matrices
- **mat_ind_final** a list of total (common + individual-specific) effect matrices
- **data** a list of multivariate time series for all subjects ...

---

**Description**

An object class to be used with cv.multivar

**Details**

To construct an object of class multivar, use the function `constructModel`

**Slots**

- `k` Numeric. The number of subjects (or groupings) in the dataset.
- `n` Numeric Vector. Vector containing the number of timepoints for each dataset.
- `d` Numeric Vector. Vector containing the number of variables for each dataset.
- `Ak` List. A list (length = k) of lagged (T-lag-horizon) by d multivariate time series.
- `bk` List. A list (length = k) of (T-lag-horizon) by d multivariate time series.
- `Hk` List. A list (length = k) of (horizon) by d multivariate time series.
- `A` Matrix. A matrix containing the lagged ((T-lag-horizon)k) by (d+dk) multivariate time series.
- `b` Matrix. A matrix containing the non-lagged ((T-lag-horizon)k) by (d) multivariate time series.
- `H` Matrix. A matrix containing the non-lagged (horizon k) by d multivariate time series.
- `lag` Numeric. The VAR order. Currently only lag 1 is supported.
- `horizon` Numeric. Forecast horizon.
- `t1` Numeric vector. Index of time series in which to start cross validation for individual k.
- `t2` Numeric vector. Index of time series in which to end cross validation for individual k.
- `lambda1` Numeric vector. Regularization parameter 1.
- `lambda2` Numeric vector. Regularization parameter 2.
- `nlambda1` Numeric. Number of lambda1 values to search over. Default is 30.
- `nlambda2` Numeric. Number of lambda2 values to search over. Default is 30.
- `tol` Numeric. Convergence tolerance.
- `depth` Numeric. Depth of grid construction. Default is 1000.
- `window` Numeric. Size of rolling window.
- `standardize` Logical. Default is true. Whether to standardize the individual data.
- `weightest` Character. Default is "mlr" for multiple linear regression. "sls" for simple linear regression also available. How to estimate the first-stage weights.
canonical Logical. Default is false. If true, individual datasets are fit to a VAR(1) model.
threshold Logical. Default is false. If true, and canonical is true, individual transition matrices are thresholded based on significance.
lassotype Character. Default is "adaptive". Choices are "standard" or "adaptive" lasso.
intercept Logical. Default is FALSE.
W Matrix. Default is NULL.
ratios Numeric vector. Default is NULL.

See Also

constructModel

Description

Simulate multivar data.

Usage

multivar_sim(
  k,
  d,
  n,
  prop_fill_com,
  prop_fill_ind,
  lb,
  ub,
  sigma,
  mat_common = NULL,
  mat_unique = NULL,
  mat_total = NULL
)

Arguments

k Integer. The number of individuals (or datasets) to be generated.
d Integer. The number of variables per dataset. For now this will be constant across individuals.
n Integer. The time series length.
prop_fill_com Numeric. The proportion of nonzero paths in the common transition matrix.
prop_fill_ind Numeric. The proportion of nonzero unique (not in the common transition matrix or transition matrix of other individuals) paths in each individual transition matrix.
plot_results

lb Numeric. The upper bound for individual elements of the transition matrices.
ub Numeric. The lower bound for individual elements of the transition matrices.
sigma Matrix. The (population) innovation covariance matrix.
mat_common Matrix. A common effects transition matrix (if known).
mat_unique List. A list of unique effects transition matrix (if known).
mat_total List. A list of total effects transition matrix (if known).

Examples

k <- 3
d <- 5
n <- 50
prop_fill_com <- .2
prop_fill_ind <- .2
lb <- 0.1
ub <- 0.7
sigma <- diag(0.1,d)
data <- multivar_sim(k, d, n, prop_fill_com, prop_fill_ind, lb, ub, sigma)$data

plot_results  Plot data arising from cv.multivar.

Description

Plot data arising from cv.multivar.

Usage

plot_results(x, plot_type = "common", facet_ncol = 3, datasets = "all")

Arguments

x Object. An object returned by multivar_sim.
plot_type Character. User can specify "common" to plot the common effects matrix, "unique" to plot the unique effects matrix, or "total" to plot the total effects matrix.
facet_ncol Numeric. Number of columns to use in the "unique" or "total" effects plot.
datasets Numeric. A vector containing the index of datasets to plot. Default is "all". 
Examples

```r
sim1 <- multivar_sim(
  k = 2, # individuals
  d = 3, # number of variables
  n = 20, # number of timepoints
  prop_fill_com = 0.1, # proportion of paths common
  prop_fill_ind = 0.1, # proportion of paths unique
  lb = 0.1, # lower bound on coefficient magnitude
  ub = 0.9, # upper bound on coefficient magnitude
  sigma = diag(1,3) # noise
)

model1 <- constructModel(data = sim1$data, weightest = "ols")
fit1 <- cv.multivar(model1)
plot_results(fit1, plot_type = "common")
```

---

**plot_sim**

Plot data arising from multivar_sim.

Description

Plot data arising from multivar_sim.

Usage

```r
plot_sim(x, plot_type = "common", facet_ncol = 3, datasets = "all")
```

Arguments

- `x` Object. An object returned by multivar_sim.
- `plot_type` Character. User can specify "common" to plot the common effects matrix, "unique" to plot the unique effects matrix, or "total" to plot the total effects matrix.
- `facet_ncol` Numeric. Number of columns to use in the "unique" or "total" effects plot.
- `datasets` Numeric. A vector containing the index of datasets to plot. Default is "all".

Examples

```r
k <- 3
d <- 5
n <- 50
prop_fill_com <- .2
prop_fill_ind <- .2
lb <- 0.1
ub <- 0.7
```
sig <- diag(0.1, d)
sim <- multivar_sim(k, d, n, prop_fill_com, prop_fill_ind, lb, ub, sigma)
plot_sim(sim, plot_type = "common")

---

**Description**

Default show method for an object of class `multivar`

**Usage**

```r
## S4 method for signature 'multivar'
show(object)
```

**Arguments**

- `object`  
  multivar object created from `ConstructModel`

**Value**

Displays the following information about the multivar object:

- To do.

**See Also**

`constructModel`
Index

* datasets
  dat_multivar_sim, 6
* multivar
  multivar_sim, 8
  plot_results, 9
  plot_sim, 10
* plot
  plot_results, 9
  plot_sim, 10
* simulate
  multivar_sim, 8
  plot_results, 9
  plot_sim, 10
* var
  multivar_sim, 8
  plot_results, 9
  plot_sim, 10

canonical.multivar, 2
canonical.multivar,multivar-method
  (canonical.multivar), 2
constructModel, 3, 3, 7, 8, 11
cv.multivar, 5
cv.multivar,multivar-method
  (cv.multivar), 5
dat_multivar_sim, 6
multivar (multivar-package), 2
multivar-class, 7
multivar-package, 2
multivar_sim, 8

plot_results, 9
plot_sim, 10

show,multivar-method (show.multivar), 11
show.multivar, 11