Package 'MNS'

December 8, 2015

Type Package

Title Mixed Neighbourhood Selection

Version 1.0

Date 2015-12-06

Author Ricardo Pio Monti, Christoforos Anagnostopoulos and Giovanni Montana

Maintainer Ricardo Pio Monti <ricardo.monti08@gmail.com>

Depends igraph, MASS, glmnet, mvtnorm, parallel, R (>= 2.10.1)

Imports doParallel

Description An implementation of the mixed neighbourhood selection (MNS) algorithm. The MNS algorithm can be used to estimate multiple related precision matrices. In particular, the motivation behind this work was driven by the need to understand functional connectivity networks across multiple subjects. This package also contains an implementation of a novel algorithm through which to simulate multiple related precision matrices which exhibit properties frequently reported in neuroimaging analysis.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2015-12-08 14:53:44

R topics documented:

`MNS-package` *Mixed Neighbourhood Selection package*

Description

An R package for estimating multiple, related grapical models using the Mixed Neighbourhood Selection algorithm. This package also includes two algorithm through which to simulate multiple, related graphical models which demonstrate some of the properties reported through empirical studies of functional connectivity networks.

Details

Package:	MNS
Type:	Package
Version:	1.0
Date:	2015-10-14
License:	GPL-2

Author(s)

Ricardo Pio Monti

References

Monti, R., Anagnostopolus, C., Montana, G. "Inferring brain connectivity networks from functional MRI data via mixed neighbourhood selection", arXiv, 2015

See Also

MNS, cv.MNS, plot.MNS, gen.Network

Examples

```
set.seed(1)
N=4
Net = gen.Network(method = "cohort", p = 10,
                       Nsub = N, sparsity = .2,
                       REsize = 20, REprob = .5,
                       REnoise = 1, Nobs = 10)
## Not run:
# plot simulated networks:
plot(Net, view="pop")

# run MNS algorithm:
```

```
mns = MNS(dat = Net$Data, lambda_pop = .1, lambda_random = .1, parallel = TRUE)

# plot results from MNS algorithm:
plot(mns) # plot population network
plot(mns, view="var") # plot variance network
plot(mns, view="sub") # plot subject networks (note red edges here are variable edges!)

## End(Not run)
```

---

cv.MNS *Select regularization parameters via cross-validation*

Description

Select regularization parameters via K-fold cross-validation

Usage

```
cv.MNS(dat, l1range, alpharange,
    K = 5, parallel = FALSE,
    cores = NULL, verbose = FALSE)
```

Arguments

`dat`	List where each entry corresponds to the time series observations for each subject
`l1range`	Vector of candidate regularization parameters. See details below
`alpharange`	Vector of candidate weighting parameters. See details below.
`K`	Number of cross-validation folds
`parallel`	Indicate whether model fit should be done in parallel. Default is FALSE
`cores`	If fit in parallel, indicate how many units/cores should be used
`verbose`	Print progress. Only available for non-parallel implementation

Details

Select regularization parameters via cross-validation. In the interest of simplicity we re-parameterize penalty as an elastic net penalty:

$$\lambda * \alpha ||\beta||_1 + \lambda * (1-\alpha) ||\sigma||_1$$

Thus λ is the regularization parameter (specified by the `l1range` argument) and α is the weighting parameter (specified by the `alpharange` argument).

Value

`l1`	selected regularization parameter
`alpha`	selected weighting parameter
`CV`	grid of cross-validation error for each pair of regularization parameters

Author(s)

Ricardo Pio Monti

References

Arlot, S., and Alain C. "A survey of cross-validation procedures for model selection." Statistics surveys 4 (2010): 40-79.

Monti, R., Anagnostopolus, C., Montana, G. "Inferring brain connectivity networks from functional MRI data via mixed neighbourhood selection", arXiv, 2015

See Also

MNS

Examples

```
set.seed(1)
Dat = gen.Network(p = 10, Nsub = 5,
 sparsity = .2, REsize=10, REprob=.5,
 REnoise = 1, Nobs=20)
## Not run:
CVs = cv.MNS(dat = Dat, l1range = seq(.1, .5, length.out=10),
 alpharange = seq(.2, .8, length.out = 5),
 parallel = FALSE, verbose = TRUE)

## End(Not run)
```

---

gen.Network   *Simulate random networks for a population of subjects*

Description

Implementations of two methods through which to simulation multiple related networks. The first simulates networks from a three-class population described in Danaher et al. (2014). The second simulates networks according to method proposed in Monti et al. (2015). For further details see the package vignette.

Usage

```
gen.Network(method = "cohort", p,
    Nobs, Nsub, sparsity,
    REsize, REprob, REnoise)
```

Arguments

`method`	Network simulation method. One of either "Danaher" for the three-class method of Danaher et al. (2014) or "cohort" for the cohort method described in Monti et al. (2015)
`p`	Number of nodes in network (i.e., this will be dimensionality of the resulting precision matrices)
`Nobs`	Number of observations per subject (assumed constant across subjects). If this is missing then only the precision matrices will be returned (i.e., random data is not simulated)
`Nsub`	Number of subjects for which to simulate networks. Note that this is set to 3 if method="Danaher"
`sparsity`	Sparsity level of precision matrices
`REsize`	Number of random effects edges to add to each subject (only for method="cohort")
`REprob`	Probability with which a random edge added to each subject (only for method="cohort")
`REnoise`	Variability of random edges (only for method="cohort")

Details

See package vignette for further details. Alternatively see Danaher et al. (2014) or Monti et al. (2015)

Value

`Networks`	List containing simulated netowrks where ith entry is the ith random network for the ith subject
`Data`	List where ith entry is simulated data for ith subject
`PopNet`	Population precision matrix (only if method="cohort")
`RanNet`	Sparse support for random edges (only if method="cohort")

Author(s)

Ricardo Pio Monti

References

Danaher, P., Wang, P. , and Witten, D. "The joint graphical lasso for inverse covariance estimation across multiple classes." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76.2 (2014): 373-397.

Monti, R., Anagnostopolus, C., Montana, G. "Inferring brain connectivity networks from functional MRI data via mixed neighbourhood selection", arXiv, 2015

See Also

`MNS`, `cv.MNS`, `plot.MNS`

Examples

```
# generate data according to cohort model of Monti et al. (2015)
set.seed(1)
Dat = gen.Network(p = 10, Nsub = 5,
 sparsity = .2, REsize=10, REprob=.5,
 REnoise = 1, Nobs=20)

## Not run:
# plot simulated networks:
plot(Net, view="pop")

## End(Not run)
```

MNS *Mixed Neighbourhood Selection*

Description

Estimate multiple related graphical models using the mixed neighbourhood selection (MNS) algorithm.

Usage

```
MNS(dat, lambda_pop, lambda_random,
    parallel = FALSE, cores = NULL,
    max_iter = 100, tol = 1e-05)
```

Arguments

`dat`	List where each entry corresponds to the time series observations for each subject
`lambda_pop`	Regularization parameter applied to fixed effects components. See details below for more information
`lambda_random`	Regularization parameter applied to the standard deviations of random effect effects. See details below for more information
`parallel`	Indicate whether model fit should be done in parallel. Default is FALSE
`cores`	If fit in parallel, indicate how many cores should be used
`max_iter`	Maximum number of iterations in EM algorithm. See details below for more information
`tol`	Convergence tolerance in EM algorithm

Details

The MNS algorithm is an extension of neighbourhood selection to the scenario where the objective is to learn multiple related Gaussian graphical models. For further details see Monti et al. (2015).

Value

PresPop	Population connectivity matrix - encodes the sparse support structure of population precision
PresRE	Network of highly variable edges - encodes the sparse support structure of highly variable edges
PresBLUP	Array containing predicted subject specific deviations from population connectivity.
it	Iterations to fit MNS model (one per node)

Author(s)

Ricardo Pio monti

References

Monti, R., Anagnostopolus, C., Montana, G. "Inferring brain connectivity networks from functional MRI data via mixed neighbourhood selection", arXiv, 2015

See Also

cv.MNS, plot.MNS

Examples

```
set.seed(1)
N=4
Net = gen.Network(method = "cohort", p = 10,
                       Nsub = N, sparsity = .2,
                       REsize = 20, REprob = .5,
                       REnoise = 1, Nobs = 10)
## Not run:
mns = MNS(dat = Net$Data, lambda_pop = .1, lambda_random = .1, parallel = TRUE)
# plot results:
plot(mns) # plot population network
plot(mns, view="var") # plot variance network
plot(mns, view="sub") # plot subject networks (note red edges here are variable edges!)


## End(Not run)
```

---

plot.MNS *Plotting function for MNS objects*

Description

Plotting function for MNS objects. This function implements plotting for either population networks, high variable networks or subject-specific networks.

Usage

```
## S3 method for class 'MNS'
plot(x, view="pop", subID=NULL, ...)
```

Arguments

`x`	MNS object
`view`	Plotting view. This argument defines which networks are plotted. Three options are "pop": plot population network, "var": plot network of variable edges, "sub": plot subject-specific networks
`subID`	If view="sub", subID indicates which subjects networks should be plotted.
`...`	Additional arguments to pass to plot function

Details

Plotting function for MNS objects. Can be used to plot simulated networks or results obtained from running MNS algorithm. Note that if networks are simulated using the `"Danaher"` method then only subject-specific networks can be plotted (i.e., we require `view="sub"`)

Author(s)

Ricardo Pio monti

References

Monti, R., Anagnostopolus, C., Montana, G. "Inferring brain connectivity networks from functional MRI data via mixed neighbourhood selection", arXiv, 2015

See Also

`MNS`, `gen.Network`

Examples

```
set.seed(1)
N=4
Net = gen.Network(method = "cohort", p = 10,
                       Nsub = N, sparsity = .2,
                       REsize = 20, REprob = .5,
                       REnoise = 1, Nobs = 10)
## Not run:
# can plot simulated networks:
plot.MNS(Net, view="pop")

# can also plot results from MNS algorithm:
mns = MNS(dat = Net$Data, lambda_pop = .1, lambda_random = .1, parallel = TRUE)
plot.MNS(mns) # plot population network
plot.MNS(mns, view="var") # plot variance network
plot.MNS(mns, view="sub") # plot subject networks (note red edges here are variable edges!)
```

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

Index