Package ‘NetworkChange’

February 6, 2020

Title Bayesian Package for Network Changepoint Analysis
Version 0.6
Date 2020-02-04 14:17:31 UTC
Author Jong Hee Park [aut,cre], Yunkyu Sohn [aut]
Maintainer Jong Hee Park <jongheepark@snu.ac.kr>
Depends R (>= 2.10.0), MCMCpack, ggplot2
Imports grid, Rmpfr, abind, mvtnorm, tidyr, sna, igraph, qgraph,
      network, stats, MASS, methods, RColorBrewer, reshape, ggrepel,
      gridExtra, rlang, GGally, ggvis
Description Network changepoint analysis for undirected network data. The package implements a hidden Markov network change point model (Park and Sohn 2019). Functions for break number detection using the approximate marginal likelihood and WAIC are also provided.
License GPL-3
NeedsCompilation no
RoxygenNote 6.1.1
Repository CRAN
Encoding UTF-8
Date/Publication 2020-02-06 05:20:02 UTC

R topics documented:

BreakDiagnostic ................................................. 2
BreakPointLoss .................................................. 3
drawPostAnalysis ............................................. 5
drawRegimeRaw ................................................. 6
kmeansU ......................................................... 7
MajorAlly ....................................................... 7
MakeBlockNetworkChange ................................. 8
MarginalCompare ........................................... 9
multiplot ..................................................... 10
Detect a break number using different metrics

Usage

\[
\text{BreakDiagnostic}(Y, R = 2, \text{mcmc} = 100, \text{burnin} = 100, \text{verbose} = 100, \text{thin} = 1, \text{UL.Normal} = \text{"Orthonormal"}, v_0 = \text{NULL}, v_1 = \text{NULL}, \text{break.upper} = 3, a = 1, b = 1)
\]

Arguments

- **Y**: Response tensor
- **R**: Dimension of latent space. The default is 2.
- **mcmc**: The number of MCMC iterations after burnin.
- **burnin**: The number of burn-in iterations for the sampler.
- **verbose**: A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the \(\beta\) vector, and the error variance are printed to the screen every verbose\(th\) iteration.
### BreakPointLoss

**thin**
The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.

**UL.Normal**
Transformation of sampled U. Users can choose "NULL", "Normal" or "Orthonormal." "NULL" is no normalization. "Normal" is the standard normalization. "Orthonormal" is the Gram-Schmidt orthogonalization. Default is "NULL."

**v0**
v0/2 is the shape parameter for the inverse Gamma prior on variance parameters for V. If v0 = NULL, a value is computed from a test run of NetworkStatic.

**v1**
v1/2 is the scale parameter for the inverse Gamma prior on variance parameters for V. If v1 = NULL, a value is computed from a test run of NetworkStatic.

**break.upper**
Upper threshold for break number detection. The default is break.upper = 3.

**a**
a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.

**b**
b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.

### References


### Examples
```r
## Not run:
set.seed(19333)
## Generate an array (15 by 15 by 20) with a block merging transition
Y <- MakeBlockNetworkChange(n=5, T=20, type = "merge")

## Fit 3 models (no break, one break, and two break) for break number detection
detect <- BreakDiagnostic(Y, R=2, break.upper = 2)

## Look at the graph
detect[[1]]; print(detect[[2]])
```

## End(Not run)

---

**BreakPointLoss**

*Compute the Average Loss of Hidden State Changes from Expected Break Points*

**Description**

Compute the Average Loss of Hidden State Changes from Expected Break Points.
BreakPointLoss

Usage

BreakPointLoss(model.list, waic = FALSE, display = TRUE)

Arguments

model.list MCMC output objects. These have to be of class mcmc and have a logmarglike attribute. In what follows, we let \( M \) denote the total number of models to be compared.

waic If waic is TRUE, waic (Watanabe information criterion) will be reported.

display If display is TRUE, a plot of ave.loss will be produced.

Value

BreakPointLoss returns five objects. They are: ave.loss the expected loss for each model computed by the mean squared distance of hidden state changes from the expected break points.

logmarglike the natural log of the marginal likelihood for each model;

State sampled state vectors;

tau expected break points for each model; and

tau.samp sampled break points from hidden state draws.

References


Examples

```r
## Not run:
set.seed(1973)
## Generate an array (30 by 30 by 40) with block transitions from 2 blocks to 3 blocks
Y <- MakeBlockNetworkChange(n=10, T=40, type ="split")
G <- 100 ## Small mcmc scans to save time

## Fit multiple models for break number detection using Bayesian model comparison
out0 <- NetworkStatic(Y, R=2, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
out1 <- NetworkChange(Y, R=2, m=1, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
out2 <- NetworkChange(Y, R=2, m=2, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
out3 <- NetworkChange(Y, R=2, m=3, mcmc=G, burnin=G, verbose=G, Waic=TRUE)

## The most probable model given break number 0 to 3 and data is out1 according to WAIC
out <- BreakPointLoss(out0, out1, out2, out3, waic=TRUE)

print(out[["ave.loss"]])
```

## End(Not run)
drawPostAnalysis

Plot of latent node cluster

Description

Plot latent node cluster

Usage

drawPostAnalysis(mcmcout, Y, point.cex = 3, text.cex = 3, segment.size = 0.1, n.cluster = NULL)

Arguments

  mcmcout  NetworkChange output
  Y        Input raw data
  point.cex node point size. Default is 3.
  text.cex node label size. Default is 3.
  segment.size segment size. Default is 0.1.
  n.cluster number of cluster. Default is 3.

Value

A plot object

References


Examples

  ## Not run:
  set.seed(1973)
  ## generate an array with two constant blocks
  data(MajorAlly)
  Y <- MajorAlly
  fit <- NetworkChange(newY, R=2, m=2, mcmc=G, initial.s = initial.s, burnin=G, verbose=0, v0=v0, v1=v1)
  drawPostAnalysis(fit, Y, n.cluster=c(4, 4, 3))

  ## End(Not run)
**drawRegimeRaw**

*Plot of network by hidden regime*

**Description**

Plot latent node cluster

**Usage**

```r
drawRegimeRaw(mcmcout, Y)
```

**Arguments**

- `mcmcout`: NetworkChange output
- `Y`: Input raw data

**Value**

A plot object

**References**


**Examples**

```r
## Not run:
set.seed(1973)
## generate an array with two constant blocks
data(MajorAlly)
Y <- MajorAlly
fit <- NetworkChange(newY, R=2, m=2, mcmc=G, initial.s = initial.s,
    burnin=G, verbose=0, v0=v0, v1=v1)
drawRegimeRaw(fit, newY)
## End(Not run)
```
## kmeansU

**Usage**

\[ \text{kmeansU(out, R = 2, n.cluster = 3, layer = 1, main = "")} \]

### Arguments

- **out**: Output of networkchange objects.
- **R**: Number of latent space dimensions
- **n.cluster**: Number of latent cluster
- **layer**: Layer id for the cluster analysis
- **main**: Title

### Value

A plot object

### Examples

```r
## Not run: set.seed(1973)
## generate an array with two constant blocks
Y <- MakeBlockNetworkChange(n=10, shape=10, T=10, type ="constant")
out0 <- NetworkStatic(Y, R=2, mcmc=10, burnin=10, verbose=10, UL.Normal = "Orthogonal")
## latent node positions
kmeansU(out0)
## End(Not run)
```

## MajorAlly

**Description**

This dataframe contains major power alliance network data from 1816 to 2012 (2 year interval).
The dataframe contains data for major power alliance network data from 1816 to 2012. Major power definition is the COW data set, which includes USA, UK, France, Germany (West Germany during 1954-1989), Austria, Italy, Russia, China, and Japan. In this data set, a defense pact (Type I), which is the highest level of military commitment, is coded as 1, and 0 otherwise.

**Source**


---

**Description**

MakeBlockNetworkChange generates a block-structured temporal data with breaks.

**Usage**

MakeBlockNetworkChange(n = 10, break.point = 0.5, base.prob = 0.05, break.point1 = 0.25, break.point2 = 0.75, type = "merge")

**Arguments**

- **n**: The number of nodes within a block. The total number of nodes is n*block.number.
- **break.point**: The point of break. 0 indicates the beginning, 0.5 indicates the middle, and 1 indicates the end.
- **base.prob**: The probability of link among non-block members.
- **block.prob**: The probability of link among within-block members.
- **shape**: The speed of breaks. The larger shape is, the faster the transition is. shape > 0 and shape < 8.
- **T**: The length of time.
- **break.point1**: The point of the first break in "merge-split" or "split-merge". Any number between 0 and 0.5 can be chosen. For example, 0 indicates the beginning, 0.25 indicates the 1/4th point, and 0.5 indicates the half point.
- **break.point2**: The point of the second break in "merge-split" or "split-merge". Any number between 0.5 and 1 can be chosen. For example, 0.5 indicates the beginning, 0.75 indicates the 3/4th point, and 1 indicates the end point.
- **type**: The type of network changes. Options are "constant", "merge", "split", "merge-split", "split-merge." If "constant" is chosen, the number of breaks is zero. If "merge" or "split" is chosen, the number of breaks is one. If either "merge-split" or "split-merge" is chosen, the number of breaks is two.
MarginalCompare

Value
output An output of MakeBlockNetworkChange contains a symmetric block-structured temporal network data set with breaks.

Description
Compare Log Marginal Likelihood

Usage
MarginalCompare(outlist)

Arguments
outlist List of NetworkChange objects

Value
A matrix of log marginal likelihoods.

References

See Also
WaicCompare
**multiplot**  
*Printing multiple ggplots in one file*

**Description**

Print multiple ggplots in one file. Slightly modified for packaging from the original version in the web.

**Usage**

```r
multiplot(..., plotlist = NULL, cols = 1, layout = NULL)
```

**Arguments**

- `...`: A list of ggplot objects separated by commas.
- `plotlist`: A list of ggplot objects
- `cols`: The number of columns.
- `layout`: A matrix specifying the layout. If present, 'cols' is ignored.

**Value**

A plot object

**Author(s)**

[http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/](http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/)

---

**NetworkChange**  
*Changepoint analysis of a degree-corrected multilinear tensor model*

**Description**

NetworkChange implements Bayesian multiple changepoint models to network time series data using a degree-corrected multilinear tensor decomposition method.

**Usage**

```r
NetworkChange(Y, R = 2, m = 1, initial.s = NULL, mcmc = 100,
              burnin = 100, verbose = 0, thin = 1, reduce.mcmc = NULL,
              degree.normal = "eigen", UL.Normal = "Orthonormal", DIC = FALSE,
              Waic = FALSE, marginal = FALSE, plotUU = FALSE, plotZ = FALSE,
              constant = FALSE, b0 = 0, B0 = 1, c0 = NULL, d0 = NULL,
              u0 = NULL, u1 = NULL, v0 = NULL, v1 = NULL, a = NULL,
              b = NULL)
```

---
Arguments

Y  Reponse tensor
R  Dimension of latent space. The default is 2.
m  Number of change point. If m = 0 is specified, the result should be the same as NetworkStatic.
initial.s  The starting value of latent state vector. The default is sampling from equal probabilities for all states.
mcmc  The number of MCMC iterations after burnin.
burnin  The number of burn-in iterations for the sampler.
verbose  A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the $\beta$ vector, and the error variance are printed to the screen every verbose iteration.
thin  The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
reduce.mcmc  The number of reduced MCMC iterations for marginal likelihood computations. If reduce.mcmc = NULL, mcmc/thin is used.
deregree.normal  A null model for degree correction. Users can choose "NULL", "eigen" or "Lsym." "NULL" is no degree correction. "eigen" is a principal eigen-matrix consisting of the first eigenvalue and the corresponding eigenvector. "Lsym" is a modularity matrix. Default is "eigen."
UL.Normal  Transformation of sampled U. Users can choose "NULL", "Normal" or "Orthonormal." "NULL" is no normalization. "Normal" is the standard normalization. "Orthonormal" is the Gram-Schmidt orthogonalization. Default is "NULL."
DIC  If DIC = TRUE, the deviation information criterion is computed.
Waic  If Waic = TRUE, the Watanabe information criterion is computed.
marginal  If marginal = TRUE, the log marginal likelihood is computed using the method of Chib (1995).
plotUU  If plotUU = TRUE and verbose > 0, then the plot of the latent space will be printed to the screen at every verbose iteration. The default is plotUU = FALSE.
plotZ  If plotZ = TRUE and verbose > 0, then the plot of the degree-corrected input matrix will be printed to the screen with the sampled mean values at every verbose iteration. The default is plotUU = FALSE.
constant  If constant = TRUE, constant parameter is sampled and saved in the output as attribute bmat. Default is constant = FALSE.
b0  The prior mean of $\beta$. This must be a scalar. The default value is 0.
B0  The prior variance of $\beta$. This must be a scalar. The default value is 1.
c0  = 0.1
D0  = 0.1
u0  $u_0/2$ is the shape parameter for the inverse Gamma prior on variance parameters for U. The default is 10.
$u_1/2$ is the scale parameter for the inverse Gamma prior on variance parameters for $U$. The default is 1.

$v_0/2$ is the shape parameter for the inverse Gamma prior on variance parameters for $V$. The default is 10.

$v_1/2$ is the scale parameter for the inverse Gamma prior on variance parameters for $V$. The default is the time length of $Y$.

$a$ is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding $a$ and $b$ values are assigned. The expected duration is the sample period divided by the number of states.

$b$ is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding $a$ and $b$ values are assigned. The expected duration is the sample period divided by the number of states.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute `Waic` that contains results of WAIC and the log-marginal likelihood of the model (`logmarglike`). The object also contains an attribute `prob.state` storage matrix that contains the probability of state $i$ for each period.

References


See Also

*NetworkStatic*

Examples

```r
## Not run:
set.seed(1973)
## Generate an array (30 by 30 by 40) with block transitions
from 2 blocks to 3 blocks
Y <- MakeBlockNetworkChange(n=10, T=40, type ="split")
G <- 100 ## Small mcmc scans to save time

## Fit multiple models for break number detection using Bayesian model comparison
out0 <- NetworkStatic(Y, R=2, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
out1 <- NetworkChange(Y, R=2, m=1, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
out2 <- NetworkChange(Y, R=2, m=2, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
out3 <- NetworkChange(Y, R=2, m=3, mcmc=G, burnin=G, verbose=G, Waic=TRUE)
outlist <- list(out0, out1, out2, out3)
```
NetworkChangeRobust

## The most probable model given break number 0 to 3 and data is out1 according to WAIC
WaicCompare(outlist)
plotU(out1)
plotV(out1)
## End(Not run)

NetworkChangeRobust  

*Changepoint analysis of a degree-corrected multilinear tensor model with t-distributed error*

### Description

NetworkChangeRobust implements Bayesian multiple changepoint models to network time series data using a degree-corrected multilinear tensor decomposition method with t-distributed error.

### Usage

```r
NetworkChangeRobust(Y, R = 2, m = 1, initial.s = NULL, mcmc = 100, burnin = 100, verbose = 0, thin = 1, degree.normal = "eigen", UL.Normal = "Orthonormal", plotUU = FALSE, plotZ = FALSE, b0 = 0, B0 = 1, c0 = NULL, d0 = NULL, n0 = 2, m0 = 2, u0 = NULL, u1 = NULL, v0 = NULL, v1 = NULL, a = NULL, b = NULL)
```

### Arguments

- **Y**  
  - Reponse tensor
- **R**  
  - Dimension of latent space. The default is 2.
- **m**  
  - Number of change point. If m = 0 is specified, the result should be the same as NetworkStatic.
- **initial.s**  
  - The starting value of latent state vector. The default is sampling from equal probabilities for all states.
- **mcmc**  
  - The number of MCMC iterations after burnin.
- **burnin**  
  - The number of burn-in iterations for the sampler.
- **verbose**  
  - A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the \( \beta \) vector, and the error variance are printed to the screen every verboseth iteration.
- **thin**  
  - The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
- **degree.normal**  
  - A null model for degree correction. Users can choose "NULL", "eigen" or "Lsym." "NULL" is no degree correction. "eigen" is a principal eigen-matrix consisting of the first eigenvalue and the corresponding eigenvector. "Lsym" is a modularity matrix. Default is "eigen."
Transformation of sampled $U$. Users can choose "NULL", "Normal" or "Orthonormal." "NULL" is no normalization. "Normal" is the standard normalization. "Orthonormal" is the Gram-Schmidt orthogonalization. Default is "NULL."

If `plotUU = TRUE` and `verbose > 0`, then the plot of the latent space will be printed to the screen at every `verbose`th iteration. The default is `plotUU = FALSE`.

If `plotZ = TRUE` and `verbose > 0`, then the plot of the degree-corrected input matrix will be printed to the screen with the sampled mean values at every `verbose`th iteration. The default is `plotUU = FALSE`.

The prior mean of $\beta$. This must be a scalar. The default value is 0.

The prior variance of $\beta$. This must be a scalar. The default value is 1.

$0.1$ The shape parameter of inverse gamma prior for $\sigma^2$.

$0.1$ The rate parameter of inverse gamma prior for $\sigma^2$.

$0.1$ The shape parameter of inverse gamma prior for $\gamma$ of Student-t distribution.

$0.1$ The rate parameter of inverse gamma prior for $\gamma$ of Student-t distribution.

$u_0/2$ is the shape parameter for the inverse Gamma prior on variance parameters for $U$. The default is 10.

$u_1/2$ is the scale parameter for the inverse Gamma prior on variance parameters for $U$. The default is 1.

$v_0/2$ is the shape parameter for the inverse Gamma prior on variance parameters for $V$. The default is 10.

$v_1/2$ is the scale parameter for the inverse Gamma prior on variance parameters for $V$. The default is the time length of $Y$.

$a$ is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding $a$ and $b$ values are assigned. The expected duration is the sample period divided by the number of states.

$b$ is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding $a$ and $b$ values are assigned. The expected duration is the sample period divided by the number of states.

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute `waic.out` that contains results of WAIC and the log-marginal likelihood of the model (`logmarglike`). The object also contains an attribute `prob.state` storage matrix that contains the probability of `state_i` for each period.


See Also

NetworkStatic

Examples

```r
## Not run:
set.seed(1973)
## Generate an array (30 by 30 by 40) with block transitions
from 2 blocks to 3 blocks
Y <- MakeBlockNetworkChange(n=10, T=40, type ="split")
G <- 100 ## only 100 mcmc scans to save time
## Fit models
out1 <- NetworkChangeRobust(Y, R=2, m=1, mcmc=G, burnin=G, verbose=G)
## plot latent node positions
plotU(out1)
## plot layer-specific network generation rules
plotV(out1)

## End(Not run)
```

NetworkStatic

Degree-corrected multilinear tensor model

Description

NetworkStatic implements a degree-corrected Bayesian multilinear tensor decomposition method

Usage

```r
NetworkStatic(Y, R = 2, mcmc = 100, burnin = 100, verbose = 0,
    thin = 1, reduce.mcmc = NULL, degree.normal = "eigen",
    UL.Normal = "Orthonormal", plotUU = FALSE, plotZ = FALSE,
    constant = FALSE, b0 = 0, B0 = 1, c0 = NULL, d0 = NULL,
    u0 = NULL, u1 = NULL, v0 = NULL, v1 = NULL, marginal = FALSE,
    DIC = FALSE, Waic = FALSE)
```
Arguments

Y  Reponse tensor
R  Dimension of latent space. The default is 2.
mcmc  The number of MCMC iterations after burnin.
burnin  The number of burn-in iterations for the sampler.
verbose  A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the \( \beta \) vector, and the error variance are printed to the screen every verbose iteration.
thin  The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
reduce.mcmc  The number of reduced MCMC iterations for marginal likelihood computations. If reduce.mcmc = NULL, mcmc/thin is used.
degree.normal  A null model for degree correction. Users can choose "NULL", "eigen" or "Lsym." "NULL" is no degree correction. "eigen" is a principal eigen-matrix consisting of the first eigenvalue and the corresponding eigenvector. "Lsym" is a modularity matrix. Default is "eigen."
UL.Normal  Transformation of sampled U. Users can choose "NULL", "Normal" or "Orthonormal." "NULL" is no normalization. "Normal" is the standard normalization. "Orthonormal" is the Gram-Schmidt orthgonalization. Default is "NULL."
plotUU  If plotUU = TRUE and verbose > 0, then the plot of the latent space will be printed to the screen at every verbose iteration. The default is plotUU = FALSE.
plotZ  If plotZ = TRUE and verbose > 0, then the plot of the degree-corrected input matrix will be printed to the screen with the sampled mean values at every verbose iteration. The default is plotUU = FALSE.
constant  If constant = TRUE, constant parameter is sampled and saved in the output as attribute bmat. Default is constant = FALSE.
b0  The prior mean of \( \beta \). This must be a scalar. The default value is 0.
B0  The prior variance of \( \beta \). This must be a scalar. The default value is 1.
c0  = 0.1
d0  = 0.1
u0  \( u_0/2 \) is the shape parameter for the inverse Gamma prior on variance parameters for U. The default is 10.
u1  \( u_1/2 \) is the scale parameter for the inverse Gamma prior on variance parameters for U. The default is 1.
v0  \( v_0/2 \) is the shape parameter for the inverse Gamma prior on variance parameters for V. The default is 10.
v1  \( v_1/2 \) is the scale parameter for the inverse Gamma prior on variance parameters for V. The default is the time length of Y.
marginal  If marginal = TRUE, the log marignal likelihood is computed using the method of Chib (1995).
DIC  If DIC = TRUE, the deviation information criterion is computed.
Waic  If Waic = TRUE, the Watanabe information criterion is computed.
Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute `Waic.out` that contains results of WAIC and the log-marginal likelihood of the model (`logmarglike`).

References


See Also

`NetworkChange`

Examples

```r
## Not run:
set.seed(1973)

## generate an array with three constant blocks
Y <- MakeBlockNetworkChange(n=10, shape=10, T=10, type ="constant")
G <- 100 ## Small mcmc scans to save time
out0 <- NetworkStatic(Y, R=2, mcmc=G, burnin=G, verbose=G)

## recovered latent blocks
Kmeans(out0, n.cluster=3, main="Recovered Blocks")

## contour plot of latent node positions
plotContour(out0)

## plot latent node positions
plotU(out0)

## plot layer-specific network connection rules
plotV(out0)

## End(Not run)
```
plotContour

*Contour plot of latent node positions*

**Description**

Draw a contour plot of latent node positions

**Usage**

```r
plotContour(OUT, main = "", k = 8, my.cols = brewer.pal(k, "Spectral"))
```

**Arguments**

- `OUT`: Output of networkchange objects.
- `main`: The title of plot
- `k`: The number of levels (nlevels in contour ()).
- `my.cols`: Color scale. Use brewer.pal() from RColorBrewer.

**Value**

A plot object

**Examples**

```r
## Not run: set.seed(1973)
## generate an array with two constant blocks
Y <- MakeBlockNetworkChange(n=10, shape=10, T=40, type="constant")
out0 <- NetworkStatic(Y, R=2, mcmc=10, burnin=10,
verbose=10, UL.Normal = "Orthonormal")
## contour plot of latent node positions
plotContour(out0)

## End(Not run)
```

plotnetarray

*Plot of network array data*

**Description**

Plot network array data

**Usage**

```r
plotnetarray(Y, n.graph = 4, node.size = 2, node.color = "brown",
edge.alpha = 0.5, edge.size = 0.2, edge.color = "grey")
```
Arguments

Y  network array data
n.graph  number of subgraphs. Default is 4.
node.size  node size. Default is 2.
node.color  node color. Default is "brown."
edge.alpha  transparency of edge. Default is 0.5.
edge.size  edge size. Default is 0.2.
edge.color  edge color. Default is "grey."

Value

A plot object

References


Examples

```r
## Not run:
set.seed(1973)
## generate an array with two constant blocks
Y <- MakeBlockNetworkChange(n=10, shape=1, T=20, type ="split")
plotnetarray(Y)
## End(Not run)
```

---

plotU  
Plot latent node positions

Description

Plot latent node positions

Usage

```r
plotU(OUT, Time = NULL, names = NULL, main = NULL,
      label.prob = 0.9)
```

Arguments

OUT    Output of networkchange objects.
Time    Starting of the time period. If NULL, 1.
names   Node names. If NULL, use natural numbers.
main    The title of plot
label.prob    Label print threshold. 0.9 is the default. Top 90% node names will be printed.
Value

A plot of latent node positions

References


Examples

```r
## Not run:
set.seed(1973)
## generate an array with two constant blocks
Y <- MakeBlockNetworkChange(n=10, shape=10, T=40, type ="constant")
out0 <- NetworkStatic(Y, R=2, mcmc=10, burnin=10,
verbose=10, UL.Normal = "Orthonormal")
## latent node positions
plotU(out0)
## End(Not run)
```

Plot layer-specific network generation rules.

Description

Plot layer-specific network generation rules.

Usage

```r
plotV(OUT, main = "", cex = 2)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>Output of networkchange objects.</td>
</tr>
<tr>
<td>main</td>
<td>The title of plot</td>
</tr>
<tr>
<td>cex</td>
<td>point size</td>
</tr>
</tbody>
</table>

Value

A plot object
Examples

```r
## Not run: set.seed(1973)
## generate an array with two constant blocks
Y <- MakeBlockNetworkChange(n=10, shape=10, T=40, type = "constant")
out0 <- NetworkStatic(Y, R=2, mcmc=10, burnin=10,
verbose=10, UL.Normal = "Orthonormal")
## latent node positions
plotV(out0)
## End(Not run)
```

---

**Postwar Ally**

*Postwar Alliance Network (1846 - 2012)*

**Description**

This dataframe contains postwar alliance network data from 1946 to 2012 (2 year interval).

**Format**

The dataframe has contains data for postwar alliance network data from 1946 to 2012 with 2 year interval. After removing disconnected components, 104 countries are included. In this data set, a defense pact (Type I), which is the highest level of military commitment, is coded as 1, and 0 otherwise.

**Source**


---

**startS**

*Sample a starting value of hidden states*

**Description**

Sample a starting value of hidden states

**Usage**

```
startS(Z, Time, m, initial.U, V, s2, R)
```
Arguments

- **Z**: Degree-corrected network array data.
- **Time**: The length of time.
- **m**: The number of breaks.
- **initial.\text{U}**: Initialized U matrix.
- **V**: Initialized V matrix.
- **s2**: Initialized error variance.
- **R**: The dimensionality of latent space.

Value

A state vector

---

**startUV**

*Starting values of U and V*

Description

Initialize starting values of U and V

Usage

```
startUV(Z, R, K)
```

Arguments

- **Z**: Degree-corrected network array data.
- **R**: The dimensionality of latent space.
- **K**: The dimensionality of Z.

Value

A list of U and V
**ULUstateSample**  
*Hidden State Sampler*

**Description**
Sample hidden states from hidden Markov multilinear model

**Usage**

```
ULUstateSample(m, s, ZMUt, s2, P, SOS.random)
```

**Arguments**
- **m**  
The number of break
- **s**  
Latent state vector
- **ZMUt**  
Z - MU
- **s2**  
Error variance
- **P**  
Transition matrix
- **SOS.random**  
Single observation state random perturbation

**Value**
A list of a state vector, state probabilities, and SOS.random.

---

**updateb**  
*Update time-constant regression parameters*

**Description**
Update time-constant regression parameters

**Usage**

```
updateb(Z, MU, s2, XtX, b0, B0)
```

**Arguments**
- **Z**  
Degree corrected response tensor
- **MU**  
Mean array
- **s2**  
Error variance
- **XtX**  
X'X
- **b0**  
Prior mean of beta
- **B0**  
Prior variance of beta
updatebm

**Update regime-changing regression parameters**

**Description**
Update regime-changing beta

**Usage**
updatebm(ns, K, s, s2, B0, p, ZU)

**Arguments**
- ns: The number of hidden states
- K: The dimensionality of Z
- s: Latent state vector
- s2: The variance of error
- B0: The prior variance of beta
- p: The rank of X
- ZU: Z - ULU

**Value**
A vector of regime-changing regression parameters

updateP

**Update transition matrix**

**Description**
Update transition matrix

**Usage**
updateP(s, ns, P, A0)

**Arguments**
- s: Latent state vector
- ns: The number of hidden states
- P: Transition matrix
- A0: Prior of transition matrix
updateS

**Value**

A transition matrix

---

**updateS**  
*A transtion matrix*

---

**Description**

Update latent states

**Usage**

`updateS(iter, s, V, m, Zb, Zt, Time, MU.state, P, s2, N.upper.tri, random.perturb)`

**Arguments**

- `iter`: iteration number
- `s`: the most recent latent states
- `V`: Network generation rules
- `m`: The number of breaks
- `Zb`: Z - b
- `Zt`: Z stacked by time
- `Time`: The length of time
- `MU.state`: UVU for each state
- `P`: Transition matrix
- `s2`: error variance
- `N.upper.tri`: The number of upper triangular elements
- `random.perturb`: If `random.perturb = TRUE` and a single state observation is found, the latent state is randomly selected by equal weights.

**Value**

A list of vectors containing latent states and their probabilities
**updates2m**  
*Update regime-specific variance*

**Description**  
Update regime-specific variance parameter

**Usage**  
`updates2m(ns, Zm, MU, c0, d0, Km)`

**Arguments**

- **ns**: The number of hidden states
- **Zm**: The regime-specific holder of Z - beta
- **MU**: The mean array.
- **c0**: Scalar shape parameter
- **d0**: Scalar scale parameter
- **Km**: Regime-specific dimensions

**Value**  
A scalar for a regime-specific variance

**updateU**  
*Update time-constant latent node positions*

**Description**  
Update time-constant latent node positions

**Usage**  
`updateU(K, U, V, R, Zb, s2, eU, iVU)`

**Arguments**

- **K**: The dimensionality of Z
- **U**: The most recent draw of latent node positions
- **V**: Layer-specific network generation rule
- **R**: The dimensionality of latent space
- **Zb**: Z - beta
- **s2**: error variance
- **eU**: The mean of U
- **iVU**: The variance of U
updateUm

Value

A matrix of time-constant latent node positions

Description

Update regime-specific latent node positions.

Usage

updateUm(ns, U, V, R, Zm, Km, ej, s2, eU, iVU, UL.Normal)

Arguments

ns  The number of latent states
U   The latent node positions
V   Layer-specific network generation rule.
R   The dimensionality of latent space
Zm  Regime-specific Z - beta
Km  The dimension of regime-specific Z.
ej  Regime indicator.
s2  The variance of error.
eU  The regime-specific mean of U.
iVU The regime-specific variance of U.
UL.Normal  Normalization method for U. "Normal" or "Orthonormal" are supported.

Value

A matrix of regime-specific latent node positions
updateV

Update layer specific network generation rules

Description

Update layer specific network generation rules

Usage

updateV(Zb, U, R, K, s2, eV, iVV, UTA)

Arguments

Zb  Z - beta.
U   The latent node positions.
R   The dimension of latent space.
K   The dimension of Z.
s2  The variance of error.
eV  The mean of V.
iVV The variance of V.
UTA Indicator of upper triangular array

Value

A matrix of layer specific network generation rules

updateVm

Update V from a change-point network process

Description

Update layer specific network generation rules from a change-point network process

Usage

updateVm(ns, U, V, Zm, Km, R, s2, eV, iVV, UTA)
**Arguments**

- **ns** The number of hidden regimes.
- **U** The latent node positions.
- **V** The layer-specific network generation rule.
- **Zm** The holder of $Z \cdot \beta$.
- **Km** The dimension of regime-specific $Z$.
- **R** The dimension of latent space.
- **s2** The variance of error.
- **eV** The mean of $V$
- **iVV** The variance of $V$
- **UTA** Indicator of upper triangular array

**Value**

A matrix of regime-specific layer specific network generation rules

**Usage**

```r
WaicCompare(outlist)
```

**Arguments**

- **outlist** List of NetworkChange objects

**Value**

Results of WAIC computation

A matrix of log marginal likelihoods.

**References**


**See Also**

`MarginalCompare`
Index

*Topic **datasets**
  MajorAlly, 7
  PostwarAlly, 21

BreakDiagnostic, 2
BreakPointLoss, 3

drawPostAnalysis, 5
drawRegimeRaw, 6

kmeansU, 7

MajorAlly, 7
MakeBlockNetworkChange, 8
MarginalCompare, 9, 29
multiplot, 10

NetworkChange, 10, 17
NetworkChangeRobust, 13
NetworkStatic, 12, 15, 15

plotContour, 18
plotnetarray, 18
plotU, 19
plotV, 20
PostwarAlly, 21

startS, 21
startUV, 22

ULUstateSample, 23
updateb, 23
updatebm, 24
updateP, 24
updateS, 25
updates2m, 26
updateU, 26
updateUm, 27
updateV, 28
updateVm, 28

WaicCompare, 9, 29