Package ‘dnc’

May 17, 2016

Type Package

Title Dynamic Network Clustering

Version 1.2

Date 2016-05-06

Author Daniel K. Sewell

Maintainer Dan Sewell <daniel-sewell@uiowa.edu>

Description Community detection for dynamic networks, i.e., networks measured repeatedly over a sequence of discrete time points, using a latent space approach.

License GPL (>= 2)

Imports Rcpp (>= 0.12.4), igraph, MCMCpack, vegan, skmeans, plot3D, plot3Drgl, BayesLogit, movMF, tcltk, stats, grDevices, graphics

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.2.3)

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-05-17 01:13:05

R topics documented:

`dnc-package` *Community detection for dynamic networks.*

Description

Dynamic network clustering/community detection using a latent space approach. Using temporal edge data, network actors are embedded onto a hypersphere and grouped based on direction.

Details

Package:	dnc
Type:	Package
Version:	1.0
Date:	2016-05-05
License:	GPL (>= 2)

This package can perform community detection on dynamic (temporal) network data observed at discrete time points. Communities are assumed fixed, but community membership may change. The main function is `dnc(. . . )` which can perform variational Bayes estimation or alternatively implement a Gibbs sampler. A `dnc` object is the output, for which there exists the following generic commands: `simulate()`, `plot()`, `print()`, and `BIC()`. Ignorable (MAR, MCAR) missing edge data can be incorporated into the Gibbs sampler.

Author(s)

Daniel K. Sewell

Maintainer: Daniel K. Sewell <daniel-sewell@uiowa.edu>

References

Sewell, D. K., and Chen, Y. (2016). Latent Space Approaches to Community Detection in Dynamic Networks. Bayesian Analysis. doi: 10.1214/16-BA1000. http://projecteuclid.org/euclid.ba/1461603847

Examples

```
## Not run:
VB5 = dnc(Y,M=5,p=3,"VB")
Gibbs5 = dnc(Y,M=5,p=3,"Gibbs")
print(VB5)
print(Gibbs5)
BIC(VB5)
BIC(Gibbs5)
plot(VB5)
plot(Gibbs5,aggregated=FALSE,plotRGL=FALSE)

## End(Not run)
```

BIC.dnc *Compute BIC as in Handcock et al. 2007*

Description

The following uses a BIC estimate of $\pi(Y, \hat{X}|M)$ to perform model selection. Note that this usage is not the typical BIC encountered in simpler contexts.

Usage

```
## S3 method for class 'dnc'
BIC(object, ...)
```

Arguments

`object`	A `dnc` object, a result of running `dnc(...)`
`...`	optional additional arguments. None are used.

Details

Rather than estimating the integrated likelihood $\pi(Y|G)$, this instead incorporates the MAP estimates of the latent positions and corresponds to $\pi(Y, \hat{X}|M)$. The BIC value returned is the following sum:

$$-2log(\pi(Y|\hat{X}, \hat{\theta_1})) + dim(\theta_1)log(\sum y_{ijt}) - 2log(\pi(\hat{X}|\hat{\theta_2})) + dim(\theta_2)\log(nT)$$

. See Sewell and Chen (2016) for more details.

Value

A scalar. Lower values are better.

References

Handcock, M. S., A.E. Raftery, and J. M. Tantrum (2007). Model-based clustering for social networks. J.R. Statist. Soc. A, 170, p. 301-354.

Examples

```
## Not run:
dncObjs = list()
BICvals = numeric(9)
for(i in 2:10){
  print(i)
  dncObjs[[i]] <- dnc(friendship,M=i,p=3,method="VB",
                      controls=list(nDraws=500,burnin=100,
                                    MaxItStg2=50,epsilonStg2=1e-15))
  BICvals[i-1] <- BIC(dncObjs[[i]])
```

```
}
plot(BICvals~c(2:10),type="b",pch=16,
     xlab="Number of communities",ylab="BIC value")
( MBest = which.min(BICvals)+1 )
abline(v=MBest,lty=2,col="blue")

## End(Not run)
```

---

dnc *Dynamic Network Clustering*

Description

Perform dynamic network clustering. Either variational Bayes or a Gibbs sampler may be implemented. Setting `M=0` performs variational Bayes with no clustering. Returns posterior parameters (if `method="VB"`) or approximate posterior samples (if `method="Gibbs"`), as well as the MAP estimates, which may be extracted through `dncObj$pm`.

Usage

```
dnc(Y,M,p=3,method="VB",init=NULL,hyperparms=NULL,Missing=NULL,
    controls=list(MaxIt=500,epsilon=1e-5,MaxItStg2=100,
                  epsilonStg2=1e-15,nDraws=10000,burnin=1000))
```

Arguments

- `Y` Dynamic network data. This should be in the form of a n x n x T array of 1's and 0's. Each slice corresponds to a single time point.
- `M` Number of communities (may be zero).
- `p` Dimension of the latent space.
- `method` Method of estimation, either "VB" for variational Bayes, or "Gibbs" for a Gibbs sampler.
- `init` *(Use of this argument is not recommended)* Initial values of the parameters. A named list containing `EOm`, `mu`, `Sig`, `Bi0g`, `Bitbar`, `Bithk`, `Er`, `Er2`, `ai2`, `bi2`, `nu`, `a3`, `b3`, `Es`, `Es2`, and `Gam`.
- `hyperparms` Hyperparameters. A named list with `cc`, `a0Star`, `b0Star`, `a2Star`, `b2Star`, `b3Star`, `GamStar`.
- `Missing` A matrix whose rows correspond to missing dyads. `Missing` should have three columns: row, column, and time (i.e., the indices for the NA's in `Y`). May be left as `NULL` if the missing dyads in `Y` are `NA`'s.
- `controls` A list of values to control the algorithm.
  - **MaxIt** The total number of iterations for the VB algorithm. Ignored if `method="Gibbs"` unless `M=0`.
  - **epsilon** Relative tolerance criteria for evaluating convergence.

MaxItStg2 The total number of iterations for the second stage initialization of the VB algorithm/Gibbs sampler. Ignored if `M=0`.

epsilonStg2 Relative tolerance criteria for evaluating convergence for the second stage initialization of the VB algorithm/Gibbs sampler. Ignored if `M=0`.

nDraws Total number of post-burn-in samples to be drawn via the Gibbs sampler. Ignored if `method="VB"`.

burnin The number of burn-in samples. Ignored if `method="VB"`.

Details

This function performs community detection according to the model

$$logit(P(Y_{ijt} = 1)) = \alpha + s_j X_{it}' X_{jt}$$

,

$$\pi(X_{it}|Z_{it} = m) = N(r_i \mathbf{u}_m, \tau^{-1} I_p)$$

While the latent positions, X_{it}'s, live in a p-dim Euclidean space, it is more natural to conceptualize these as living on a (hyper-) sphere with the magnitude of the X_{it}'s as attached attributes that reflect the actors' individual tendency to send and receive edges.

If `M=0`, then the prior on X_{it} is given by

$$\pi(X_{i1}) = N(0, \sigma^2 I_p)$$

$$\pi(X_{it}|X_{i(t-1)}) = N(X_{i(t-1)}, \tau_i^{-1} I_p)$$

The variational Bayes approach is typically faster than the Gibbs sampler, but tends to underestimate the spread of the posterior.

Currently, only VB is implemented when `M=0` (no clustering), hence `method` will be ignored if `M=0`.

Ignorable missing data can be estimated within the Gibbs sampler (not using the VB algorithm) by adding the extra step of drawing the missing edges given the latent positions and the model parameters at each iteration.

Using the `init` is, in general, strongly discouraged, as this may have a non-negligible negative affect on the performance of the VB or the length of the chain needed to reach convergence. Unless otherwise specified, both the initialization scheme and the hyperparameters are chosen according to Sewell and Chen (2016).

Value

An object of class `dnc`, for which other methods exist (e.g., `methods(class="dnc")`).

If `method="VB"` and `M=0`,

method The estimation algorithm

Y The original data

mu A p x T x n array: Posterior mean of the latent positions

Sig A (Tp) x p x n array: Posterior covariance matrices of the latent positions. The covariance matrix for X_{it} is `dncObj$Sig[(t-1)*p,,i]`

a0 Scalar: Posterior shape parameter for σ^2 in inverse gamma distribution (if `M=0`).

b0 Scalar: Posterior scale parameter for σ^2 in inverse gamma distribution (if `M=0`).

ai1 A n x 1 vector: Posterior mean parameter for the r_i's in truncated normal distribution (if `M>0`).

bi1 A n x 1 vector: Posterior variance parameter for the r_i's in truncated normal distribution (if `M>0`).

Er A n x 1 vector: Posterior first moment for the r_i's (if `M>0`).

Er2 A n x 1 vector: Posterior second moment for the r_i's (if `M>0`).

ai2 A n x 1 vector: Posterior shape parameter for the τ_i's in gamma distribution.

bi2 A n x 1 vector: Posterior scale parameter for the τ_i's in gamma distribution.

a3 Scalar: Posterior mean for α.

b3 Scalar: Posterior variance for α.

ai4 A n x 1 vector: Posterior mean parameter for the s_j's in truncated normal distribution.

bi4 A n x 1 vector: Posterior variance parameter for the s_j's in truncated normal distribution.

Es A n x 1 vector: Posterior first moment for the s_j's.

Es2 A n x 1 vector: Posterior second moment for the s_j's.

nu A M x p matrix: Posterior mean directions for the `M` clusters/communities, i.e., for the u_m's (if `M>0`).

kappa A M x 1 vector: Posterior concentration parameters for the `M` clusters/communities, i.e., for the u_m's (if `M>0`).

Z A n x T matrix: Cluster assignments based on the maximum posterior probabilities, computed marginally at each time point (if `M>0`).

Bi0g A n x M matrix: Posterior probabilities of community assignment for each actor at the first observed time point (if `M>0`).

Bithk A (MT) x M x n array: Posterior transition probability matrices; $\pi(Z_{itk} = 1|Z_{i(t-1)h} = 1, Y) =$ `dncObj$Bithk[(t-1)*M+h,k,i]`. Ignore first `M` lines (internal use only). (if `M>0`).

Bitbar A T x M x n array: Marginal posterior probabilities of community assignments, i.e., $\pi(Z_{itk} = 1|Y) =$ `dncObj$Bitbar[t,k,i]` (if `M>0`).

Gam A (M+1) x M matrix: Posterior concentration parameters for β_0 (row 1) and for $\beta_m, m > 1$ (rows 2 to M+1) in Dirichlet distribution (if `M>0`).

If `method="Gibbs"`,

method The estimation algorithm

Y The original data

X A p x T x n x `nDraws` array: Posterior samples for the latent positions.

r A n x `nDraws` matrix: Posterior samples for the r_i's.

tau A n x `nDraws` matrix: Posterior samples for the τ_i's.

alpha A `nDraws` x 1 vector: Posterior samples for α.

s A n x `nDraws` matrix: Posterior samples for the τ_i's.

u A M x p x `nDraws` array: Posterior draws for the communities, i.e., the u_m's.

Z A n x T x `nDraws` array: Posterior draws for the community assignments for each actor at each time point.

beta A (M+1) x M x n array: Posterior draws for $beta_0$ (row 1) and $\beta_m, m > 1$ (rows 2 to M+1).

posterior A (`burnin+nDraws`) x 1 vector: Posterior values for all iterations of the Gibbs sampler.

Missing A matrix of four columns: The row, column, and time for each missing dyad, as well as the posterior probability that the dyad equals one.

Additionally, each `dnc` class object comes with a `$pm` value, which is a list of the MAP estimates for `alpha`, `X`, `s`, `tau`, `r`, `u`, `Z`, and `beta`.

References

Sewell, D. K., and Chen, Y. (2016). Latent Space Approaches to Community Detection in Dynamic Networks. Bayesian Analysis. doi: 10.1214/16-BA1000. http://projecteuclid.org/euclid.ba/1461603847

Examples

```
data(friendship)
set.seed(123)
dncObj <- dnc(friendship,M=4,p=3,method="Gibbs",
              controls=list(nDraws=250,burnin=50,
                            MaxItStg2=25,epsilonStg2=1e-15))
print(dncObj)
BIC(dncObj)
par(mar=rep(0,4)+0.05)
plot(dncObj,plotRGL=FALSE,pch=16,phi=60,lwd=2,cex=1.5)
```

`dncMAP` *MAP estimates*

Description

(Re-)Compute the maximum a posteriori (MAP) estimates for a `dnc` object. Primarily intended as an internal function called from `dnc(...)`.

Usage

```
dncMAP(dncObj)
```

Arguments

`dncObj`	A dnc class object

Value

A list with the following elements:

alpha Intercept

X Latent positions

s Receiver scaling effects

tau Latent precision parameters

sigma2 Initial latent position variance (if `M=0` only)

r Magnitude of mean latent positions (if `M>0` only)

u Unit vectors defining communities (if `M>0` only)

Z Community assignments (if `M>0` only)

beta Initial cluster assignment probabilities (row 1) and transition probability matrix (rows 2 to `M`) (if `M>0` only)

`friendship` *Longitudinal classroom friendship network (Andrea Knecht)*

Description

This friendship network consists of 26 students in a middle school in the Netherlands measured at four time points throughout a school year.

Usage

```
data("friendship")
```

Format

The data is a 26 x 26 x 4 array of 1's and 0's (and NA's).

Details

See `knecht {xergm.common}` for more details, as well as for behavioral and demographic actor attributes.

References

Knecht, Andrea (2006): Networks and Actor Attributes in Early Adolescence [2003/04]. Utrecht, The Netherlands Research School ICS, Department of Sociology, Utrecht University. (ICS-Codebook no. 61).

Knecht, Andrea (2008): Friendship Selection and Friends' Influence. Dynamics of Networks and Actor Attributes in Early Adolescence. PhD Dissertation, University of Utrecht. http://dspace.library.uu.nl/bitstream/handle/1

Examples

```
data(friendship)
```

Internals — *Internal dnc functions*

Description

Internal functions called by dnc

Details

These functions are c++ not intended to be called by the user.

plot.dnc — *Plot a* dnc *object.*

Description

Construct 2D or 3D plots of the latent actor positions using the plot3D and plot3Drgl packages.

Usage

```
## S3 method for class 'dnc'
plot(x,aggregated=TRUE,plotRGL=TRUE,
                    Lines=TRUE,colByComm=TRUE,
                    INDEX=1:min(dim(x$pm$X)[1],3),...)
```

Arguments

x	A dnc class object
aggregated	Logical. Should the time points be aggregated, or should separate plots be made for each time point?
plotRGL	Logical. Should rgl interactive plot(s) be made? Ignored if length(INDEX)==2.
Lines	Logical. Should lines be drawn from the origin to the center of the communities?
colByComm	Logical. Should the actors be colored according to community membership?
INDEX	Vector. Which subset of the p dimensions should be plotted? **Note: The length of this vector determines whether a 2D or 3D plot should be made.**
...	Further arguments to be passed into points(), surf3D or points3D functions.

Details

This function utlizies functions from the plot3D and plot3Drgl packages for 3D functionality. To see what options you may pass into ..., see documentation for points in base R, plot3D::surf3D, or plot3D::points3D.

Examples

```
data(friendship)
set.seed(123)
dncObj <- dnc(friendship,M=4,p=3,method="Gibbs",
              controls=list(nDraws=250,burnin=50,
                            MaxItStg2=25,epsilonStg2=1e-15))
par(mar=rep(0,4)+0.05)
plot(dncObj,plotRGL=FALSE,pch=16,phi=60,lwd=2,cex=1.5)
```

---

print.dnc *Print* dnc *object*

Description

Print description of a dnc object.

Usage

```
## S3 method for class 'dnc'
print(x,printDens=FALSE,...)
```

Arguments

x	A dnc class object.
printDens	Logical. Should the density of the network at each time point be computed and printed?
...	additional optional arguments.

Examples

```
data(friendship)
set.seed(123)
dncObj <- dnc(friendship,M=4,p=3,method="Gibbs",
              controls=list(nDraws=250,burnin=50,
                            MaxItStg2=25,epsilonStg2=1e-15))
print(dncObj)
```

simulate.dnc *Simulate DNC*

Description

Use an existing dnc object to simulate new network data.

Usage

```
## S3 method for class 'dnc'
simulate(object,nsim=1,seed=NULL,...)
```

Arguments

object	A dnc class object
nsim	Integer. Number of simulations to perform.
seed	Optional numeric. If is.null(seed) is FALSE then seed is used in a call to set.seed before simulating the network data. The default, NULL, will not change the random generator state.
...	additional optional arguments.

Details

This function uses the posterior mean of the latent positions and model parameters to simulate new network data sets.

Value

A n x n x T x nsim array.

Examples

```
    data(friendship)
    set.seed(123)
    dncObj <- dnc(friendship,M=4,p=3,method="Gibbs",
                  controls=list(nDraws=250,burnin=50,
                                MaxItStg2=25,epsilonStg2=1e-15))
  newSims = simulate(dncObj,10,123)
```

Index