Package 'BLSM'

July 21, 2025

Title Bayesian Latent Space Model

Version 0.1.0

Date 2018-04-25

Description Provides a Bayesian latent space
model for complex networks, either weighted or unweighted.
Given an observed input graph, the estimates for the latent coordinates
of the nodes are obtained through a Bayesian MCMC algorithm.
The overall likelihood of the graph depends on a fundamental probability
equation, which is defined so that ties are more likely to exist
between nodes whose latent space coordinates are close.
The package is mainly based on the model by Hoff, Raftery and Handcock (2002)
<doi:10.1198/016214502388618906> and contains some extra features
(e.g., removal of the Procrustean step, weights implemented as
coefficients of the latent distances, 3D plots).
The original code related to the above model was retrieved from
<https://www.stat.washington.edu/people/pdhoff/Code/hoff_raftery_handcock_2002_jasa/>.
Users can inspect the MCMC simulation, create and customize insightful
graphical representations or apply clustering techniques.

Depends R (>= 3.3.0)

Imports Rcpp (>= 0.12.4)

Suggests rgl (>= 0.98.1)

LinkingTo Rcpp, RcppEigen

License GPL (>= 2)

LazyData true

RoxygenNote 6.0.1

NeedsCompilation yes

Author Alberto Donizetti [aut, cre],
Francesca Ieva [ctb]

Maintainer Alberto Donizetti <albe.donizetti@gmail.com>

Repository CRAN

Date/Publication 2018-04-26 11:04:13 UTC

Contents

`alpha_up` *Update step for the α variable*

Description

Accept/reject the proposal for the α model variable

Usage

```
alpha_up(Y, lpz, W, alpha, adelta, a_a, a_b)
```

Arguments

`Y`	Adjacency matrix of the observed network
`lpz`	Matrix containing the negative square root of the Euclidean distances between latent positions
`W`	BLSM Weights matrix of the observed network
`alpha`	Model variable α
`adelta`	The uniform proposal for α is defined on the $[-adelta, +adelta]$ interval
`a_a`	Shape parameter of the Gamma prior distribution for α. The value is usually set to 1, so the prior is actually an exponential distribution.
`a_b`	Rate parameter of the Gamma prior distribution for α.

Value

Updated value of the α variable

BLSM *Bayesian Latent Space Model*

Description

R package allowing the computation of a Bayesian Latent Space Model for complex networks.

Latent Space Models are characterized by the presence of unobservable variables (latent coordinates) that are used to compute the likelihood of the observed networks. Their goal is to map the observed network in the latent space by meeting specific probabilistic requirements, so that the estimated latent coordinates can then be used to describe and characterize the original graph.

In the BSLM package framework, given a network characterized by its adjacency Y matrix, the model assigns a binary random variable to each tie: Y_ij is related to the tie between nodes i and j and its value is 1 if the tie exists, 0 otherwise.

The model assumes the independence of $Y_ij|x_i, x_j, \alpha$, where x_i and x_j are the coordinates of the nodes in the multidimensional latent space and α is an additional parameter such that $logit(P(Y_ij = 1)) = \alpha - ||x_i - x_j||$.

The latent space coordinates are estimated by following a MCMC procedure that is based on the overall likelihood induced by the above equation.

Due to the symmetry of the distance, the model leads to more intuitive outputs for undirected networks, but the functions can also deal with directed graphs.

If the network is weighted, i.e. to each tie is associated a positive coefficient, the model's probability equation becomes $logit(P(Y_ij = 1)) = \alpha - W_ij||x_i - x_j||$, where W_ij denotes the weight related to link existing between x_i and x_j. This means that even non existing links should have a weight, therefore the matrix used in the computation isn't the original weights matrix but actually a specific "BLSM weights" matrix that contains positive coefficients for all the possible pairs of nodes. When dealing with weighted networks, please be careful to pass a "BLSM weights" matrix as input #' (please refer to example_weights_matrix for more detailed information and a valid example).

The output of the model allows the user to inspect the MCMC simulation, create insightful graphical representations or apply clustering techniques to better describe the latent space. See estimate_latent_positions or plot_latent_positions for further information.

References

P. D. Hoff, A. E. Raftery, M. S. Handcock, Latent Space Approaches to Social Network Analysis, Journal of the American Statistical Association, Vol. 97, No. 460, (2002), pp. 1090-1098.

A. Donizetti, A Latent Space Model Approach for Clustering Complex Network Data, Master's Thesis, Politecnico di Milano, (2017).

dst *Geodesic distance*

Description

Evaluate geodesic distance (shortest path) between all pairs of nodes in the network.

Usage

```
dst(M)
```

Arguments

M	Input adjacency matrix

Value

Matrix containing all the pairwise geodesic distances

Examples

```
dst(example_adjacency_matrix)
```

estimate_latent_positions *BLSM simulation*

Description

Core function of the BLSM package: run a simulation to obtain the positions of the network nodes in the latent space for each sampled iteration.

The positions are simulated accordingly to the model assumptions, please refer to BLSM for further information. The output of the function can be used to retrieve and compare specific iterations, observe their evolution or simply compute the average positions (more details in the descriptions and examples below).

Usage

```
estimate_latent_positions(Y, W, procrustean = TRUE, k = 2, alpha = 2,
  nscan = 8 * 10^5, burn_in = 5 * 10^5, odens = 10^3, zdelta = 1,
  z_norm_prior_mu = 0, z_norm_prior_sd = 10, adelta = 0.3,
  a_exp_prior_a = 1, a_exp_prior_b = 1, dynamic_plot = FALSE,
  dynamic_circles = FALSE, ...)
```

Arguments

`Y`	Adjacency matrix of the network
`W`	(Optional) BLSM Weight matrix of the network
`procrustean`	Boolean to include/exclude (`TRUE/FALSE`) the Procrustean Transform step in the algorithm. Set `TRUE` by default.
`k`	Space dimensionality
`alpha`	Starting value of the α variable
`nscan`	Number of iterations
`burn_in`	Burn-in value (starting iterations to be discarded)
`odens`	Thinning: only 1 iteration every `odens` will be sampled and stored in the output
`zdelta`	Standard deviation of the Gaussian proposal for latent positions
`z_norm_prior_mu`	Mean of the Gaussian prior distribution for latent positions
`z_norm_prior_sd`	Standard deviation of the Gaussian prior distribution for latent positions
`adelta`	The uniform proposal for α is defined on the $[-adelta, +adelta]$ interval
`a_exp_prior_a`	Shape parameter of the Gamma prior distribution for α. As the value is usually set to 1 the prior is an exponential distribution.
`a_exp_prior_b`	Rate parameter of the Gamma prior distribution for α.
`dynamic_plot`	Boolean to plot dynamically the simulated positions (one update every `odens` iterations)
`dynamic_circles`	Boolean to add circles of radius α to the dynamic plots
`...`	Additional parameters that can be passed to plot_latent_positions

Value

Returns a "BLSM object" (`blsm_obj`), i.e. a list containing:

- `Alpha` α values from the sampled iterations
- `Likelihood` Log-likelihood values from the sampled iterations
- `Iterations` Latent space coordinates from the sampled iterations. Latent positions are stored in a 3D array whose dimensions are given by (1: number of nodes, 2: space dimensionality, 3: number of iterations). In the non-Procrustean framework the latent distances are given instead of the positions: another 3D array is returned, whose dimensions are given by (1: number of nodes, 2: number of nodes, 3: number of iterations). The command needed in order to get the average values over the iterations for either the positions or the distances is `rowMeans(blsm_obj$Iterations, dims=2)` (see example below).
- `StartingPositions` Latent space coordinates right after the initialization step. In the non-Procrustean framework starting distances are given instead.
- `Matrix` Original matrices of the network (adjacency and BLSM weights)
- `Parameters` List of parameters specified during the call to estimate_latent_positions

Examples

```
## Not run:
 # Procrustean version followed by clustering
 blsm_obj = estimate_latent_positions(example_adjacency_matrix,
                          burn_in = 3*10^4, nscan = 10^5, dynamic_plot = TRUE)

 avg_latent_positions = rowMeans(blsm_obj$Iterations, dims=2)
 h_cl = hclust(dist(avg_latent_positions), method="complete")
 n = 3
 latent_space_clusters = cutree(h_cl, k=n)
 print(latent_space_clusters)
 plot(avg_latent_positions, col=rainbow(n)[latent_space_clusters], pch=20)

 # Non-Procrustean version followed by clustering
 blsm_obj_2 = estimate_latent_positions(example_adjacency_matrix, procrustean=FALSE,
                          burn_in = 3*10^4, nscan = 10^5)
 avg_latent_distances = rowMeans(blsm_obj_2$Iterations, dims=2)
 h_cl = hclust(as.dist(avg_latent_distances), method="complete")
 n = 3
 latent_space_clusters_2 = cutree(h_cl, k=n)
 print(latent_space_clusters_2)

 # Weighted network
 blsm_obj_3 = estimate_latent_positions(example_adjacency_matrix, example_weights_matrix,
                          burn_in = 10^5, nscan = 2*10^5, dynamic_plot = TRUE)

## End(Not run)
```

---

example_adjacency_matrix

Example Adjacency Matrix

Description

Adjacency matrix of a 10 nodes random network for testing purposes

Usage

```
example_adjacency_matrix
```

Format

A binary adjacency matrix representing links between nodes.

`example_blsm_obj` *Example BLSM object*

Description

BLSM object obtained by applying the Procrustean version of the latent space model to the unweighted network whose adjacency matrix is example_adjacency_matrix. Further details concerning the simulation are contained in the BLSM object itself.

Usage

```
example_blsm_obj
```

Format

BLSM object (`blsm_obj`), i.e. a list containing:

- `Alpha` α values from the sampled iterations
- `Likelihood` Log-likelihood values from the sampled iterations
- `Iterations` Latent space coordinates from the sampled iterations. Latent positions are stored in a 3D array whose dimensions are given by (1: number of nodes, 2: space dimensionality, 3: number of iterations). In the non-Procrustean framework the latent distances are given instead of the positions: another 3D array is returned, whose dimensions are given by (1: number of nodes, 2: number of nodes, 3: number of iterations). The command needed in order to get the average values over the iterations for either the positions or the distances is `rowMeans(blsm_obj$Iterations, dims=2)` (see example below).
- `StartingPositions` Latent space coordinates right after the initialization step. In the non-Procrustean framework starting distances are given instead.
- `Matrix` Original matrices of the network (adjacency and BLSM weights)
- `Parameters` List of parameters specified during the call to estimate_latent_positions

`example_weights_matrix` *Example Weights Matrix*

Description

"BLSM weights" matrix of a 10 nodes random network for testing purposes

Usage

```
example_weights_matrix
```

Format

A matrix containing positive weights for all pairs of nodes.

Given a couple of nodes, a weight expresses the importance of the distance between the coordinates associated to the two nodes in the latent space in terms of the overall likelihood of the graph. For this reason, even missing links must have a coefficient, otherwise the relative positioning of disconnected nodes would have no effect at all on the graph likelihood.

The exact probability equation is described in BLSM, as well as the notation used.

A few examples:

- for unweighted networks, the "BLSM weights" matrix has all the values set to 1.
- if two nodes share a strong connection, then the weight coefficient should be greater than 1 so that their positions in the latent space will be closer than they would be in an unweighted framework.
- if two nodes share a weak connection, a coefficient smaller than 1 will allow the latent coordinates to be pretty far from each other even though the nodes are connected.

`lpY` *Network log-likelihood*

Description

Compute the log-likelihood of the whole observed network based on the latent positions estimates and the model assumptions. See BLSM for more information.

Usage

```
lpY(Y, lpz, alpha, W)
```

Arguments

`Y`	Adjacency matrix of the observed network
`lpz`	Matrix containing the negative square root of the Euclidean distances between latent positions (output of lpz_dist)
`alpha`	Model variable α
`W`	BLSM Weights matrix of the observed network

Value

Log-likelihood of the observed network

lpYNODE *Network log-likelihood for individual updates*

Description

Compute the log-likelihood of the whole observed network based on the latent positions estimates and the model assumptions. The function follows almost the same approach as lpY, but it is more suitable for the individual updates occurring during the simulation.

Usage

```
lpYNODE(Y, Z, alpha, node, diag, W)
```

Arguments

Y	Adjacency matrix of the observed network
Z	Latent positions matrix
alpha	Model variable α
node	Specific node in the network corresponding to the latent coordinate which will be used as reference
diag	Diagonal from t(Z)%*%Z matrix, passed to speed up the process.
W	BLSM Weights matrix of the observed network

Value

Log-likelihood of the observed network

lpz_dist *Distance between latent positions*

Description

Compute the square root of the Euclidean distances between latent positions and return them with a negative sign.

Usage

```
lpz_dist(Z)
```

Arguments

Z	Latent positions matrix. The matrix size must be (n,k), where n and k denote respectively the number of nodes in the network and the latent space dimensionality.

Value

Matrix containing the negative square root of the Euclidean distances between latent positions

Examples

```
pos = matrix(rnorm(20), ncol=2)
lpz_dist(pos)
```

`lpz_distNODE` *lpz_dist optimized for individual updates*

Description

Compute the square root of the Euclidean distances between a specific coordinate in the latent space and all the others. The function follows almost the same approach as lpz_dist, but it is more suitable for the individual updates occurring during the simulation.

Usage

```
lpz_distNODE(Z, node, diag)
```

Arguments

`Z`	Latent positions matrix
`node`	Specific node in the network corresponding to the latent coordinate which will be used as reference
`diag`	Diagonal from `t(Z)%*%Z` matrix, passed to speed up the process.

Value

Vector containing the negative square root of the Euclidean distances between latent positions

---

`mlpY` *Network (positive) log-likelihood*

Description

Compute the (positive) log-likelihood of the whole observed network based on the latent positions estimates and the model assumptions. The inputs are slightly different from those of lpY, so the function basically applies some preprocessing before calling lpY and returning its value with the opposite sign.

Usage

```
mlpY(avZ, Y, W)
```

Arguments

avZ	Vector containing the α value and the latent positions
Y	Adjacency matrix of the observed network
W	BLSM Weights matrix of the observed network

Value

Log-likelihood of the observed network

---

plot_latent_positions *Base BLSM plot function*

Description

Plot latent positions from a Procrustean simulation.

Usage

```
plot_latent_positions(blsm_obj, colors, points_size = 0.1,
  labels_point_size = 5, labels_point_color = "yellow",
  labels_text_size = 1, labels_text_color = "blue", circles_2D = FALSE)
```

Arguments

blsm_obj	BLSM object obtained through estimate_latent_positions
colors	(Optional) Colors of the simulated coordinate points in the latent space. Internal default colors are used if the argument is missing.
points_size	Size of the coordinate points
labels_point_size	Size of the label points
labels_point_color	Color of the label points
labels_text_size	Text size in the label points
labels_text_color	Text color in the label points
circles_2D	Plot circles of radius α (see the model's main variables) centered around the label points

Examples

```
plot_latent_positions(example_blsm_obj, labels_point_color = "black", labels_text_color = "black")

plot_latent_positions(example_blsm_obj, circles_2D = TRUE)
```

plot_traceplots_acf *BLSM traceplots and ACF*

Description

Traceplots and autocorrelation functions for the α variable and a selected node (or pair of nodes in the non-Procrustean framework).

Usage

```
plot_traceplots_acf(blsm_obj, chosen_node = 1, coordinate = 1,
  chosen_pair = c(1, 2))
```

Arguments

blsm_obj	BLSM object obtained through estimate_latent_positions
chosen_node	Specified node for traceplot and autocorrelation function (Procrustean framework)
coordinate	Specified coordinate dimension from the n-dimensional latent space
chosen_pair	Specified pair of nodes for traceplot and autocorrelation function (non-Procrustean framework)

Examples

```
plot_traceplots_acf(example_blsm_obj, chosen_node=3, coordinate=1)

## Not run:
 # Run the simulation without Procrustean step
 blsm_obj = estimate_latent_positions(example_adjacency_matrix, procrustean = FALSE,
                          burn_in = 3*10^4, nscan = 10^5)

 # Plot
 plot_traceplots_acf(blsm_obj, chosen_pair=c(2,5))

## End(Not run)
```

proc_crr *Procrustean corresponding positions*

Description

Given a set of starting coordinates, the function returns the Procrustean Transform of the initial points that minimizes the sum of squared positional difference from a set of reference coordinates. The (Euclidean) distances between a candidate configuration and the reference are evaluated by considering the couples of corresponding points.

The reference configuration must be centered at the origin.

Usage

```
proc_crr(Z, Z0)
```

Arguments

Z	set of initial coordinates to be transformed
Z0	set of reference coordinates centered at the origin

Value

Set of coordinates minimizing the distance between the initial configuration and the reference one

Examples

```
# Create configuration and center it at the origin
pos_ref = matrix(runif(20), ncol=2)
pos_ref = t(t(pos_ref)-colMeans(pos_ref))

# Create a new configuration by adding a perturbation to the previous one
pos = pos_ref + matrix(rnorm(20, mean=1, sd=0.1), ncol=2)

# Compute the Procrustean Transform and inspect the results
proc_pos = proc_crr(pos, pos_ref)
plot(pos_ref, col="blue", pch=20, xlim=c(-1,3), ylim=c(-1,3))
points(pos, col="red", pch=20)
points(proc_pos, col="purple", pch=20)
```

Z_up *Update step for the latent positions*

Description

Accept/reject the proposals for the latent positions

Usage

```
Z_up(Y, Z, W, alpha, zdelta, mu_z, sd_z)
```

Arguments

Y	Adjacency matrix of the observed network
Z	Latent positions matrix
W	BLSM Weights matrix of the observed network
alpha	Model variable α
zdelta	Standard deviation of the Gaussian proposal for latent positions
mu_z	Mean of the Gaussian prior distribution for latent positions
sd_z	Standard deviation of the Gaussian prior distribution for latent positions

Value

Updated latent positions matrix

Index