Package 'RandClust'

July 25, 2020

Type Package

Title Randomized spectral clustering for large-scale networks

Version 0.1.0

Author Who wrote it

Maintainer The package maintainer <yourself@somewhere.net>

Description
This package implements spectral clustering for large-scale directed and undirected networks using randomization techniques including the random projection and the random sampling.

License GPL

Encoding UTF-8

LazyData true

Imports Rcpp,
RSpectra,
irlba,
stats,
RcppZiggurat,
Matrix,
Gmedian,
igraph

LinkingTo Rcpp, RcppEigen

SystemRequirements C++11

RoxygenNote 7.1.0

R topics documented:

`rclust` *Randomized spectral clustering using random sampling or random projection*

Description

Randomized spectral clustering for undirected networks. The clusters are computed using two random schemes, namely, the random sampling and the random projection scheme. Can deal with very large networks.

Usage

```
rclust(
  A,
  method = c("rsample", "rproject"),
  k,
  p = 10,
  q = 2,
  dist = "normal",
  P,
  iter.max = 50,
  nstart = 10,
  ...
)
```

Arguments

`A`	The adjacency matrix of an undirected network with type "dgCMatrix".
`method`	The method for computing the randomized eigendecomposition. Random sampling-based eigendecomposition is implemented if `method="rsample"`, and random projection-based eigendecomposition is implemented if `method="rproject"`.
`k`	The number of target clusters.
`p`	The oversampling parameter in the random projection scheme. Requested only if `method="rproject"`. Default is 10.
`q`	The power parameter in the random projection scheme. Requested only if `method="rproject"`. Default is 2.
`dist`	The distribution of the entry of the random test matrix in the random projection scheme. Requested only if `method="rproject"`. Default is `"normal"`.
`P`	The sampling probability in the random sampling scheme. Requested only if `method="rsample"`.
`iter.max`	Maximum number of iterations in the `kmeans`. Default is 50.
`nstart`	The number of random sets in `kmeans`. Default is 10.
`...`	Additional arguments.

Details

This function computes the clusters of undirected networks using randomized spectral clustering algorithms. The random projection-based eigendecomposition or the random sampling-based eigendecomposition is first computed for the adjacency matrix of the undirected network. The k-means is then performed on the randomized eigen vectors.

Value

`cluster`	The cluster vector (from `1:k`) with the numbers indicating which cluster each node is assigned.
`rvectors`	The randomized k eigen vectors computed by `reig.pro` or `reig.sam`.

See Also

`reig.pro`, `reig.sam`.

Examples

```
n <- 100
k <- 2
clustertrue <- rep(1:k, each = n/k)
A <- matrix(0, n, n)
for(i in 1: (n-1)) {
   for(j in (i+1):n) {
       A[i, j] <- ifelse(clustertrue[i] == clustertrue[j], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
       A[j, i] <- A[i, j]
    }
}
A <- as(A, "dgCMatrix")
rclust(A, method = "rsample", k = k, P = 0.7)
```

---

rcoclust *Randomized spectral co-clustering using random sampling or random projection*

Description

Randomized spectral co-clustering for directed networks. The row clusters and column clusters are computed using two random schemes, namely, the random sampling and the random projection scheme. Can deal with very large networks.

Usage

```
rcoclust(
  A,
  method = c("rsample", "rproject"),
  ky,
  kz,
  p = 10,
  q = 2,
```

```
  dist = "normal",
  P,
  normalize = FALSE,
  iter.max = 50,
  nstartkmedian = 10,
  nstartkmeans = 10,
  ...
)
```

Arguments

A	The adjacency matrix of a directed network with type "dgCMatrix".
method	The method for computing the randomized SVD. Random sampling-based SVD is implemented if method="rsample", and random projection-based SVD is implemented if method="rproject".
ky	The number of target row clusters.
kz	The number of target column clusters.
p	The oversampling parameter in the random projection scheme. Requested only if method="rproject". Default is 10.
q	The power parameter in the random projection scheme. Requested only if method="rproject". Default is 2.
dist	The distribution of the entry of the random test matrix in the random projection scheme. Requested only if method="rproject". Default is "normal".
P	The sampling probability in the random sampling scheme. Requested only if method="rsample".
normalize	The k-means is implemented if FALSE. Otherwise the spherical k-median is implemented. Default is FALSE.
iter.max	Maximum number of iterations in the kmeans and that for choosing the starting point of kGmedian. Default is 50.
nstartkmedian	The number of times the k-median algorithm is ran in the kGmedian. Default is 10.
nstartkmeans	The number of random sets in kmeans and that for choosing the starting point of kGmedian. Default is 10.
...	Additional arguments.

Details

This function computes the row clusters and column clusters of directed networks using randomized spectral co-clustering algorithms. The random projection-based SVD or the random sampling-based SVD is first computed for the adjacency matrix of the directed network. The k-means or the spherical k-median is then performed on the randomized singular vectors.

Value

cluster.y	The row cluster vector (from 1:ky) with the numbers indicating which row cluster each node is assigned.
cluster.z	The column cluster vector (from 1:kz) with the numbers indicating which column cluster each node is assigned.

rvectors.y	The randomized left min(ky,kz) singular vectors computed by rsvd.pro or rsvd.sam.
rvectors.z	The randomized right min(ky,kz) singular vectors computed by rsvd.pro or rsvd.sam.

See Also

rsvd.pro, rsvd.sam.

Examples

```
n <- 120
ky <- 2
kz <- 3
cluster.y <- rep(1:ky, each = n/ky)
cluster.z <- rep(1:kz, each = n/kz)
probmat <- matrix(0.2, ky, kz)
diag(probmat) <- 0.1
A <- sample_scbm(type = "scbm", cluster.y, cluster.z, probmat = probmat, graph = FALSE)
rcoclust(A, method ="rsample", ky, kz, P = 0.7, normalize = FALSE)
rcoclust(A, method ="rproject", ky, kz, normalize = TRUE)
```

---

reig.pro *Compute randomized eigenvalue decomposition of a symmetric matrix using random projection*

Description

Compute the randomized eigenvalue decomposition of a symmetric matrix by random projection. The randomized eigen vectors and eigen values are computed. Can deal with very large data matrix.

Usage

```
reig.pro(A, rank, p = 10, q = 2, dist = "normal", approA = FALSE)
```

Arguments

A	Input data matrix of class "dgCMatrix". The matrix should be a symmetric matrix but not necessarily be the adjacency matrix of a network.
rank	The target rank of the low-rank decomposition.
p	The oversampling parameter. It need to be a positive integer number. Default value is 10.
q	The power parameter. It need to be a positive integer number. Default value is 2.
dist	The distribution of the entry of the random test matrix. Can be "normal" (standard normal distribution), "unif" (uniform distribution from -1 to 1), or "rademacher" (randemacher distribution). Default is "normal".
approA	A logical variable indicating whether the approximated A is returned. Default is FALSE.

Details

This function computes the randomized eigen value decomposition of a data matrix using the random projection scheme. The data matrix A is symmetric. It is first compressed to a smaller matrix with its columns (rows) being the linear combinations of the columns (rows) of A. The classical eigen value decomposition is then performed on the smaller matrix. The randomized eigen value decomposition of A are obtained by postprocessing.

Value

vectors	The randomized rank+p eigen vectors.
values	The rank+p eigen values.
approA	The approximated data matrix if requested.

References

N. Halko, P.-G. Martinsson, and J. A. Tropp. (2011) *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, *SIAM review, Vol. 53(2), 217-288*
https://epubs.siam.org/doi/10.1137/090771806

Examples

```
n <- 100
rank <- 2
clustertrue <- rep(1:rank, each = n/rank)
A <- matrix(0, n, n)
for(i in 1:(n - 1)) {
    for(j in (i + 1):n) {
        A[i, j] <- ifelse(clustertrue[i] == clustertrue[j], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
    }
}
diag(A) <- 0
A <- A + t(A)
A <- as(A, "dgCMatrix")
reig.pro(A, rank)
```

---

reig.sam *Compute randomized eigenvalue decomposition of a matrix using random sampling*

Description

Compute the randomized eigenvalue decomposition of a matrix by random sampling. The randomized eigen vectors and eigen values are computed. Can deal with very large data matrix.

Usage

```
reig.sam(A, P, use_lower = TRUE, k, tol = 1e-05, ...)
```

Arguments

A	An input sparse data matrix of type "dgCMatrix". A is not necessarily a symmetric matrix, see the parameter use_lower.
P	The sampling probability. Should be between 0 and 1.
use_lower	If TRUE/FALSE, only the lower/upper triangular part of A is used for sampling and the following eigendecomposition steps.
k	Number of eigen values requested.
tol	Precision parameter of the iterative algorithm. Default is 1e-5.
...	Additional arguments of function eigs.

Details

This function computes the randomized eigenvalue decomposition of a data matrix using the random sampling scheme. The data matrix A is first sampled to obtain a sparsified matrix. An iterative algorithm (eigs) for computing the leading eigen vectors is then performed on the sparsified matrix to obtain the randomized eigen vectors and eigen values.

Value

vectors	The randomized k eigen vectors.
values	The k eigen values.
sparA	The sparsified data matrix obtained by rsample_sym(A,P).

See Also

rsample_sym, eigs.

Examples

```
n <- 100
k <- 2
clustertrue <- rep(1:k, each = n/k)
A <- matrix(0, n, n)
for(i in 1:(n-1)) {
    for(j in (i+1):n) {
        A[i, j] <- ifelse(clustertrue[i] == clustertrue[j], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
        A[j, i] <- A[i, j]
    }
}
diag(A) <- 0
A <- as(A, "dgCMatrix")
reig.sam(A, P = 0.7, use_lower = TRUE, k = k)
```

rsample *Sample a sparse matrix*

Description

Sample a sparse matrix

Usage

```
rsample(A, P)
```

Arguments

A	A sparse matrix of type "dgCMatrix".
P	The probability that each edge is kept.

Value

A binary sparse matrix of type "dgCMatrix".

Examples

```
library(Matrix)
set.seed(123)
n = 20
A = matrix(rbinom(n^2, 1, 0.5), 20, 20)
diag(A) = 0
A = as(A, "dgCMatrix")
A
rsample(A, 0.5)
```

rsample_sym *Sample a symmetric sparse matrix*

Description

Sample a symmetric sparse matrix

Usage

```
rsample_sym(A, P, use_lower = TRUE)
```

Arguments

A	A sparse matrix of type "dgCMatrix". A does not need to be symmetric, see the parameter use_lower.
P	The probability that each edge is kept.
use_lower	If TRUE/FALSE, only the lower/upper triangular part of A is used for sampling.

Value

A lower triangular, binary, and sparse matrix of type "dgCMatrix". The diagonal elements are all zeros.

Examples

```
library(Matrix)
set.seed(123)
n = 20
A = matrix(rbinom(n^2, 1, 0.5), 20, 20)
A = as(A, "dgCMatrix")
A
rsample_sym(A, 0.5, use_lower = TRUE)
rsample_sym(A, 0.5, use_lower = FALSE)
```

---

rsvd.pro *Compute randomized SVD of a matrix using random projection*

Description

Compute the randomized SVD of a matrix by random projection. The randomized singular vectors and singular values are computed. Can deal with very large data matrix.

Usage

```
rsvd.pro(A, rank, p = 10, q = 2, dist = "normal", approA = FALSE, nthread = 1)
```

Arguments

A	Input data matrix of class "dgCMatrix". Not necessarily be the adjacency matrix of a network.
rank	The target rank of the low-rank decomposition.
p	The oversampling parameter. It need to be a positive integer number. Default value is 10.
q	The power parameter. It need to be a positive integer number. Default value is 2.
dist	The distribution of the entry of the random test matrix. Can be "normal" (standard normal distribution), "unif" (uniform distribution from -1 to 1), or "rademacher" (randemacher distribution). Default is "normal".
approA	A logical variable indicating whether the approximated A is returned. Default is FALSE.
nthread	Maximum number of threads for specific computations which could be implemented in parallel. Default is 1.

Details

This function computes the randomized SVD of a data matrix using the random projection scheme. The data matrix A is first compressed to a smaller matrix with its columns (rows) being the linear combinations of the columns (rows) of A. The classical SVD is then performed on the smaller matrix. The randomized SVD of A are obtained by postprocessing.

Value

u	The randomized left rank+p singular vectors.
v	The randomized right rank+p singular vectors.
d	The rank+p singular values.
approA	The approximated data matrix obtained by udv' if requested.

References

N. Halko, P.-G. Martinsson, and J. A. Tropp. (2011) *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, *SIAM review, Vol. 53(2), 217-288*
https://epubs.siam.org/doi/10.1137/090771806

Examples

```
library(Matrix)
n <- 100
rank <- 2
clustertrue.y <- rep(1:rank, each = n/rank)
clustertrue.z <- rep(1:rank, each = n/rank)
A <- matrix(0, n, n)
for(i in 1:n) {
     for(j in 1:n) {
        A[i, j] <- ifelse(clustertrue.y[i] == clustertrue.z[i], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
    }
}
diag(A) <- 0
A <- as(A, "dgCMatrix")
rsvd.pro(A, rank)
```

---

rsvd.sam *Compute randomized SVD of a matrix using random sampling*

Description

Compute the randomized SVD of a matrix by random sampling. The randomized singular vectors and singular values are computed. Can deal with very large data matrix.

Usage

```
rsvd.sam(A, P, k = max(nu, nv), nu, nv, tol = 1e-05, ...)
```

Arguments

A	An input sparse data matrix of type "dgCMatrix". Not necessarily be the adjacency matrix of a network.
P	The sampling probability. Should be between 0 and 1.

k	Number of singular values requested. k should not be smaller than nu or nv. The default value is max(nu,nv).
nu	Number of left singular vectors to be computed.
nv	Number of right singular vectors to be computed.
tol	Precision parameter of the iterative algorithm. Default is 1e-5.
...	Additional arguments of function irlba.

Details

This function computes the randomized SVD of a data matrix using the random sampling scheme. The data matrix A is first sampled to obtain a sparsified matrix. An iterative algorithm (irlba) for computing the leading singular vectors is then performed on the sparsified matrix to obtain the randomized singular vectors and singular values.

Value

u	The randomized left nu singular vectors.
v	The randomized right nv singular vectors.
d	The k leading singular values.
sparA	The sparsified data matrix obtained by rsample(A,P).

See Also

rsample, irlba.

Examples

```
n <- 100
rank <- 2
clustertrue.y <- rep(1:rank, each = n/rank)
clustertrue.z <- rep(1:rank, each = n/rank)
A <- matrix(0, n, n)
for(i in 1:n) {
    for(j in 1:n) {
        A[i, j] <- ifelse(clustertrue.y[i] == clustertrue.z[i], rbinom(1, 1, 0.2), rbinom(1, 1, 0.1))
    }
}
diag(A) <- 0
A <- as(A, "dgCMatrix")
rsvd.sam(A, P = 0.7, nu = rank, nv = rank)
```

---

sample_scbm *Generate directed networks from (degree-corrected) stochastic co-block models.*

Description

Produce directed networks or their adjacency matrices from (degree-corrected) stochastic co-block models.

Usage

```
sample_scbm(
  type = c("scbm", "dc-scbm"),
  cluster.y,
  cluster.z,
  theta.y,
  theta.z,
  probmat,
  graph = FALSE
)
```

Arguments

type	If type="scbm", the network is generated from the stochastic co-block model. If type="dc-scbm", the network is generated from the degree-corrected stochastic co-block model.
cluster.y	The row cluster vector (from 1:k) with the numbers indicating which row cluster each node is assigned.
cluster.z	The column cluster vector (from 1:k) with the numbers indicating which row cluster each node is assigned. cluster.z and cluster.y must have the same length but not necessarily indicate the same clusters and cluster numbers.
theta.y	The vector indicating the propensity of each node to send edges. Suggested value is from 0 to 1. Only required when type="dc-scbm".
theta.z	The vector indicating the propensity of each node to receive edges. Suggested value is from 0 to 1. Only required when type="dc-scbm".
probmat	The link probability matrix with dimension c(max(cluster.y,cluster.z)). probmat[i,j] is proportional to the probability of an edge from nodes in row cluster i to nodes in column cluster j.
graph	If TRUE, an igraph object is returned. Otherwise, an adjacency matrix is returned. Default is FALSE.

Details

This function generates directed networks using stochstic co-block models and degree-corrected stochastic co-block models. In the stochastic co-block models, nodes in the same row (column) block are stochastic equivalent senders (receivers) in the sense that two nodes send out (receive) an edge to (from) a third node with the same probability if these two nodes are in the same row (column) cluster. In the degree-corrected stochastic co-block model, the probability of an edge also depends on the nodes propensity.

Value

A	The adjacency matrix of generated networks. Only returned if graph=FALSE.
g	An igraph object. Only returned if graph=TRUE.

References

K. Rohe, T. Qin, and B. Yu. (2016) *Co-clustering directed graphs to discover asymmetries and directional communities*, *Proceedings of the National Academy of Sciences, Vol. 113(45), 12679-12684*
https://www.pnas.org/content/113/45/12679

Examples

```
# The four parameter stochastic co-block model
n <- 100
ky <- 2
kz <- 2
cluster.y <- rep(1:ky, each = n/ky)
cluster.z <- rep(1:kz, each = n/kz)
probmat <- matrix(0.01, ky, ky)
diag(probmat) <- 0.1
g1 <- sample_scbm(type = "scbm", cluster.y, cluster.z, probmat = probmat, graph = TRUE)
plot(g1, vertex.size = 8, edge.arrow.size=0.3, vertex.label = NA, vertex.color = c("green3", "cyan3")[clust
vertex.frame.color = NA)

# The degree-corrected stochastic co-block model
n <- 50
ky <- 2
kz <- 2
cluster.y <- rep(1:ky, each = n/ky)
cluster.z <- rep(1:kz, each = n/kz)
probmat <- matrix(0.5, ky, kz)
diag(probmat) <- 0.8
theta.y <- rep(0.8,n)
theta.y [c(1,n)] <- 1
theta.z <- rep(1,n)
g2 <- sample_scbm(type = "dc-scbm", cluster.y, cluster.z, theta.y, theta.z, probmat, graph = TRUE)
plot(g2, vertex.size = 8, edge.arrow.size=0.3, vertex.label = NA, vertex.color = c("red", "purple")[cluster
vertex.frame.color = NA)
```

---

scoEpinions *Epinions social network*

Description

This is a who-trust-whom online social network of a general consumer review site. Members of the site can decide whether to ”trust” each other. All the trust relationships interact and form the Web of Trust which is then combined with review ratings to determine which reviews are shown to the user.

Usage

```
data(scoEpinions)
```

Format

The `scoEpinions` object is a sparse matrix representing the adjacency matrix of the Epinions social network.

Details

The largest connected component of the original network is collected. There is 75877 nodes and 508836 edges.

Source

http://snap.stanford.edu/data/soc-Epinions1.html

References

M. Richardson, R. Agrawal, and P. Domingos. (2003) *Trust management for the semantic web*, *International semantic Web conference, 351-368*

Examples

```
data(scoEpinions)
A <- scoEpinions
rsvd.sam(A, P = 0.7, nu = 3, nv = 3)
```

---

youtubeNetwork *Youtube social network*

Description

This is a Youtube social network where users form friendship each other.

Usage

```
data(youtubeNetwork)
```

Format

The youtubeNetwork object is a sparse matrix representing the adjacency matrix of the Youtube social network.

Details

There is 1134890 nodes and 2987624 edges.

Source

http://snap.stanford.edu/data/com-Youtube.html

References

J. Yang and J. Leskovec. (2012) *Defining and Evaluating Network Communities based on Ground-truth*, *ICDM*

Examples

```
data(youtubeNetwork)
A <- youtubeNetwork
reig.sam(A, P=0.7, k = 4)
reig.pro(A, rank = 4)
```

Index