Package 'PropClust'

July 21, 2025

Type Package

Title Propensity Clustering and Decomposition

Version 1.4-7

Date 2023-09-06

Author John Michael O Ranola, Kenneth Lange, Steve Horvath, Peter Langfelder

Maintainer Peter Langfelder <Peter.Langfelder@gmail.com>

Depends R (>= 3.0.0), fastcluster, dynamicTreeCut

Imports stats

Description Implementation of propensity clustering and
decomposition as described in Ranola et al. (2013) <doi:10.1186/1752-0509-7-21>.
Propensity decomposition can be viewed on the
one hand as a generalization of the eigenvector-based
approximation of correlation networks, and on the other hand as
a generalization of random multigraph models and
conformity-based decompositions.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-10-06 12:40:05 UTC

Contents

CPBADecomposition *Cluster and Propensity-based Approximation decomposition for adjacency matrixes.*

Description

Given an adjacency matrix and cluster assignments, this function calculates either the conformity factors or the propensities of each node.

Usage

```
CPBADecomposition(adjacency,
                  clustering,
                  nClusters = NULL,
                  objectiveFunction = c("Poisson", "L2norm"),
                  dropUnassigned = TRUE,
                  unassignedLabel = 0,
                  unassignedMethod = "average",
                  accelerated = TRUE,
                  parallel = FALSE)
```

Arguments

`adjacency`	A square symmetric matrix giving either the number of connections between two nodes (for Poisson objective function) or the weighted connections (between 0 and 1) between each pair of nodes.
`clustering`	A vector with element per node containing the cluster assignments for each node. If a single cluster decomposition is desired, an alternative is to set `nClusters=1` (see below).
`nClusters`	If the user wishes to input trivial clustering to calculate a "pure propensity" decomposition, this variable can be set to 1. Any other non-NULL value is considered invalid; use `clusters` to specify a non-trivial clustering.
`objectiveFunction`	Specifies the objective function for the Cluster and Propensity-based Approximation. Valid choices are (unique abbreviations of) "Poisson" and "L2norm".
`dropUnassigned`	Logical: should unassigned nodes be excluded from the clustering? Unassigned nodes can be present in initial clustering or blocks (if given), and internal pre-partitioning and initial clustering can also lead to unassigned nodes. If `dropUnassigned` is `TRUE`, these nodes are excluded from the calls to `propensityClustering`. Otherwise these nodes will be assigned to the nearest cluster within each block and be clustered using `propensityClustering` in each block.
`unassignedLabel`	Label in input `clustering` that is reserved for unassigned objects. For clusterings with numeric lables this is typically (but not always) 0. Note that this must a valid value - missing value `NA` will not work.

`unassignedMethod`
: If `dropUnassigned` is `FALSE`, this argument sepcifies the method to assign unassigned objects to the nearest cluster. Valid values are (unique abbreviations) of "average", "single", and "complete". In analogy with hierarchical clustering, each node will be assigned to the cluster with which it has the highest average, maximum, and minimum adjacency, respectively.

`accelerated`
: Logical: should an accelerated algorithm be used? In general the accelerated method is preferable.

`parallel`
: Logical: should parallel calculation be used? At present the parallel calculation is not fully implemented and the function falls back to standard accelerated calculation, with a warning.

Details

If a single cluster is specified, the approximation is known as "Pure Propensity".

If unassigned nodes are present in the clustering and they are dropped before the CPBA calculation, their propensities, mean values and tail p-values are returned as NA.

Value

Returns the following list of items.

`Propensity`
: Gives the propensities (or conformities) of each node.

`IntermodularAdjacency`
: Gives the intermodular adjacencies or the conformities between clusters.

`Factorizability`
: Gives the factorizability of the data.

`L2Norm` or `Loglik`
: The L2 Norm (for L2 norm objective function) or the log-likelihood (for Poisson objetive function).

`ExpectedAdjancency`
: A distance structure representing the lower triangle of the symmetric matrix of estimated values of the adjacency matrix using the Propensity and IntermodularAdjacency. If the Poisson updates are used, the returned values are the estimate means of the distribution.

`EdgePvalues`
: A distance structure representing the lower triangle of the symmetric matrix of the tail probabilities under the Poisson distribution.

Author(s)

John Michael Ranola, Peter Langfelder, Steve Horvath, Kenneth Lange

References

Ranola et. al. (2010) A Poisson Model for Random Multigraphs. Bioinformatics 26(16):2004-2001.
Ranola JM, Langfelder P, Lange K, Horvath S (2013) Cluster and propensity based approximation of a network. BMC Bioinformatics, in press.

See Also

propensityClustering

Examples

```
nNodes=50
nClusters=5
#We would like to use L2Norm instead of Loglikelihood
objective = "L2norm"

ADJ<-matrix(runif(nNodes*nNodes),ncol=nNodes)
for(i in 1:(length(ADJ[1,])-1)){
for(j in i:length(ADJ[,1])){
ADJ[i,j]=ADJ[j,i]
}
}

for(i in 1:length(ADJ[1,])) ADJ[i,i]=0

Results<-propensityClustering(
              adjacency = ADJ,
              objectiveFunction = objective,
              initialClusters = NULL,
              nClusters = nClusters,
              fastUpdates = FALSE)

Results2<-CPBADecomposition(adjacency = ADJ, clustering = Results$Clustering,
                            objectiveFunction = objective)

Results3<-propensityClustering( adjacency = ADJ,
              objectiveFunction = objective,
              initialClusters = NULL,
              nClusters = nClusters,
              fastUpdates = TRUE)
```

---

propensityClustering *Propensity clustering*

Description

This function performs propensity clustering that assigns objects (or nodes) in a network to clusters such that the resulting Cluster and Propensity-based Approximation (CPBA) of the input adjacency matrix optimizes a specific criterion. Large data sets on which standard propensity clustering may take too long are first optionally split into smaller blocks. Propensity clustering is then applied to each block, and the clustering is used for the final CPBA decomposition.

Usage

```
propensityClustering(
  adjacency,
  decompositionType = c("CPBA", "Pure Propensity"),
  objectiveFunction = c("Poisson", "L2norm"),
  fastUpdates = TRUE,
  blocks = NULL,
  initialClusters = NULL,
  nClusters = NULL,
  maxBlockSize = if (fastUpdates) 5000 else 1000,
  clustMethod = "average",
  cutreeDynamicArgs = list(deepSplit = 2, minClusterSize = 20,
                           verbose = 0),
  dropUnassigned = TRUE,
  unassignedLabel = 0,
  verbose = 2,
  indent = 0)
```

Arguments

`adjacency`	Adjacency matrix of the network: a square, symmetric, non-negative matrix giving the connection strengths between pairs of nodes. Missing data are not allowed.
`decompositionType`	Decomposition type. Either the full CPBA (Cluster and Propensity-Based Approximation) or pure propensity, which is a special case of CPBA when all nodes are in a single cluster.
`objectiveFunction`	Objective function. Available choices are `"Poisson"` and `"L2norm"`.
`fastUpdates`	Logical: should a fast, "approximate", propensity clustering method be used? This option is recommended unless the number of nodes to be clustered is small (less than 500). The fast updates may lead to slightly inferior results but are orders of magnitude faster for larger data sets (above say 500 nodes).
`blocks`	Optional specification of blocks. If given, must be a vector with length equal the number of columns in `adjacency`, each entry giving the block label for the corresponding node. If not given, blocks will be determined automatically.
`initialClusters`	Optional specification of initial clusters. If given, must be a vector with length equal the number of columns in `adjacency`, each entry giving the cluster label for the corresponding node. If not given, initial clusters will be determined automatically. The method depends on whether `nClusters` (see below) is specified.
`nClusters`	Optional specification of the number of clusters. Note that specifying `nClusters` changes the cluster initialization method. If nodes are split into blocks, the number of clusters in each block will equal `nClusters`, and the total number of clusters will be `nClusters` times the number of blocks.
`maxBlockSize`	Maximum block size.

`clustMethod` Hierarchical clustering method. Recognized options are "average", "complete", and "single".

`cutreeDynamicArgs` Arguments (options) for the `cutreeDynamic` function from package `dynamicTreeCut` used in the initial clustering step. Arguments `dendro` and `distM` are set automatically; the rest can be set by the user to fine-tune the process of initial cluster identification.

`dropUnassigned` Logical: should unassigned nodes be excluded from the clustering? Unassigned nodes can be present in initial clustering or blocks (if given), and internal pre-partitioning and initial clustering can also lead to unassigned nodes. If `dropUnassigned` is `TRUE`, these nodes are excluded from the calls to `propensityClustering`. Otherwise these nodes will be assigned to the nearest cluster within each block and be clustered using `propensityClustering` in each block.

`unassignedLabel` Label in input `blocks` and `initialClustering` that is reserved for unassigned objects. For clusterings with numeric lables this is typically (but not always) 0. Note that this must a valid value - missing value `NA` will not work.

`verbose` Level of verbosity of printed diagnostic messages. 0 means silent (except for progress reports from the underlying propensity clustering function), higher values will lead to more detailed progress messages.

`indent` Indentation of the printed diagnostic messages. 0 means no indentation, each unit adds two spaces.

Details

If `initialClusters` are not given, they are determined from the adjancency in one of the following two ways: if `nClusters` is not specified, the initialization uses hierarchical clustering followed by the Dynamic Tree Cut (see `cutreeDynamic`). Arguments and options for the `cutreeDynamic` can be specified using the argument `cutreeDynamicArgs`. Some nodes may be left unassigned and their handling is described below. If `nClusters` is specified, an internal initialization algorithm based on connectivities is used. This second algorithm assigns all nodes to a cluster.

If `dropUnassigned` is `TRUE`, nodes left unassigned by the clustering procedure are excluded from the following calculations. If `dropUnassigned` is `FALSE`, nodes left unassigned by the clustering procedure are assigned to their nearest cluster, using the clustering dissimilarity measure specified in `clustMethod`.

In the next step, if the total number of nodes exceeds maximum block size, the initial clusters (either given or those automatically determined by hierarchical clustering) are split into blocks. Clusters bigger than maximum block size `maxBlockSize` are put into separate blocks (one cluster per block). Clusters smaller than maximum block size are placed into blocks such that the block size does not exceed `maxBlockSize` and such that clusters with high between-cluster adjacency are placed in the same block, if possible. The between-cluster adjacency is consistent with `clustMethod`.

Note that for the purposes of splitting data into blocks, hierarchical clustering is always used. If the internal initialization of clusters is used, it is applied within each block and idependently of all other blocks.

Next, propensity clustering is applied to each block. More precisely, propensity clustering is applied to the subset of nodes in each block that is assigned to an initial cluster. Some nodes may not be assigned to initial clusters and these nodes are excluded from propensity clustering.

Once propensity clustering on all blocks is finished, propensity decomposition is calculated on the entire network (excluding unassigned nodes).

Value

List with the following components:

`Clustering` The final clustering. A vector of length equal to the number of nodes (columns in `adjacency`) givig the cluster labels for each node. Clusters are labeled 1,2,3,... Label 0 is reserved for unassigned nodes.

`Propensity` Propensities (or conformities) of each node.

`NodeWasConsidered`
Logical vector with one entry per node. `TRUE` if the node was part of the propensity clustering and decomposition (recall that unassigned nodes are excluded).

`IntermodularAdjacency`
Intermodular adjacencies or the conformities between clusters.

`Factorizability`
Factorizability of the data.

`L2Norm` or `Loglik`
The L2 Norm or the loglikelihood depending on l2bool.

`MeanValues` A distance structure representing the lower triangle of the symmetric matrix of estimated values of the adjacency matrix using the Propensity and IntermodularAdjacency. If the Poisson updates are used, the returned values are the estimate means of the distribution.

`TailPvalues` A distance structure representing the lower triangle of the symmetric matrix of the tail probabilities under the Poisson distribution.

`Blocks` Blocks. A vector with one component for each node giving the block label for each node. The blocks are labeled 1,2,3,...

`InitialClusters`
The initial clusters. A copy of the input if given, otherwise the automatically determined initial clutering.

`InitialTree` The hierarchical clustering dendrogram (tree) used to determine initial clusters. Only present if the initial clusters were not supplied by the user.

Author(s)

John Michael Ranola, Peter Langfelder, Kenneth Lange, Steve Horvath

References

Ranola et. al. (2010) A Poisson Model for Random Multigraphs. Bioinformatics 26(16):2004-2001.
Ranola JM, Langfelder P, Lange K, Horvath S (2013) Cluster and propensity based approximation of a network. MC Syst Biol. 2013 Mar 14;7:21. doi: 10.1186/1752-0509-7-21.

See Also

CPBADecomposition for propensity decomposition;

hclust for the hierarchical clustering function,

cutreeDynamic for the dynamic tree cut to identify clusters in a dendrogram

Examples

```
# Simulate 50 nodes in 5 clusters
nNodes=50
nClusters=5
# We would like to use L2Norm instead of Loglikelihood
objective = "L2norm"

ADJ<-matrix(runif(nNodes*nNodes),ncol=nNodes)

ADJ = (ADJ + t(ADJ))/2;

diag(ADJ) = 0;

results<-propensityClustering(
              adjacency = ADJ,
              objectiveFunction = objective,
              initialClusters = NULL,
              nClusters = nClusters,
              fastUpdates = FALSE)

table(results$Clustering)
```

Index