Package 'HyperG'

May 7, 2026

Type Package

Title Hypergraphs in R

Version 1.0.0

Date 2021-03-01

Author David J. Marchette

Maintainer David J. Marchette <dmarchette@gmail.com>

Description Implements various tools for storing and analyzing hypergraphs. Handles basic undirected, unweighted hypergraphs, and various ways of creating hypergraphs from a number of representations, and converting between graphs and hypergraphs.

Depends R (>= 3.1.0), igraph, mclust

Imports proxy, RSpectra, gtools, grDevices, Matrix, stats, graphics

Suggests testthat (>= 3.0.0)

License GPL (>= 2)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2021-03-04 09:20:11 UTC

Contents

HyperG-package *Hypergraphs in R*

Description

Implements various tools for storing and analyzing hypergraphs. Handles basic undirected, unweighted hypergraphs, and various ways of creating hypergraphs from a number of representations, and converting between graphs and hypergraphs.

Details

A hypergraph is implemented as a list containing (for now) a single element, `M`, corresponding to the incidence matrix. It is an S3 object with class `hypergraph` and a plot method, summary and print methods. The package uses a sparse representation (from the **Matrix** package), so in principle it should allow for very large hypergraphs, although to date only relatively small hypergraphs have been investigated.

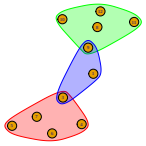
Index of help topics:

```
H2                      Two sections of a hypergraph.
HyperG-package          Hypergraphs in R
as.bipartite            Hypergraph as a bipartite graph.
as.hypergraph           Convert between hypergraphs and graphs.
ase                     Adjacency spectral embedding.
clique_hypergraph       Clique Hypergraph
cluster_spectral        Spectral Graph Clustering
dual_hypergraph         Dual hypergraph.
epsilon_hypergraph      Epsilon-Ball Hypergraph
equivalent.hypergraphs
                        Equivalent Hypergraphs
has.helly               Helly Property
has.isolates            Test for loops, isolates and empty hyper-edges.
hdegree                 Degrees of a hypergraph.
horder                  The number of vertices, edges and statistics of
                        the hypergraph.
hrank                   Rank of a hypergraph.
hypergraph.add.edges    Add edges or vertices to a hypergraph.
hypergraph.complement   The complement of a hypergraph.
hypergraph.delete.edges
                        Delete edges or vertices of a hypergraph.
hypergraph.entropy      Hypergraph Entropy
hypergraph.is.connected
                        Is the hypergraph connected?
hypergraph.union        Unions and intersections of hypergraphs.
hypergraph_as_adjacency_matrix
                        Adjacency Matrix of a Hypergraph.
hypergraph_as_edgelist
```

```
                        Convert between hypergraphs and graphs.
hypergraph_from_incidence_matrix
                        Hypergraph construction.
hypergraph_from_literal
                        Hypergraph from literal.
hypergraph_laplacian_matrix
                        Laplacian Matrix
incidence_matrix        Graph Incidence Matrix.
induced_hypergraph      Induced hypergraph.
is.conformal            Conformal Hypergraphs
is.empty.hypergraph     Is the hypergraph empty.
is.hypergraph           Is an object a hypergraph?
is.hypertree            Test for hypertree.
is.simple               Is a hypergraph simple/linear?
is.star                 Is a hypergraph a star?
is.tree                 Test if a graph is a tree or a forest.
kCores                  K-Cores
knn_hypergraph          K-Nearest Neighbor Hypergraph.
line.graph              Line Graph
make_empty_hypergraph   Empty hypergraph.
pendant                 Pendant Vertices
plot.hypergraph         Plot a hypergraph.
print.hypergraph        Print a hypergraph to the console.
reduce.hypergraph       Remove redundant hyperedges and isolated
                        vertices.
remove.redundant.vertices
                        Remove redundant vertices.
reorder_vertices        Reorder the vertices of a hypergraph.
sample_geom_hypergraph
                        Construct a hypergraph from a random collection
                        of points.
sample_gnp_hypergraph   Erdos-Renyi hypergraphs.
sample_k_uniform_hypergraph
                        Random k-uniform and k-regular hypergraphs.
sample_sbm_hypergraph   Sample from a stochastic block model.
subtree.hypergraph      Subtree Hypergraph.
summary.hypergraph      Print a summary of the hypergraph to the
                        console.
```

Introduction

A graph is a set of vertices, V, and a set of egdes, E, each of which contains two vertices (or a single vertex, if self-loops are allowed). A hypergraph is a generalization of this, in which more than two vertices can be in a single hyper-edge. Multi-graphs are graphs in which E is not a set, but rather allows for duplicate edges. Hypergraphs are allowed to have duplicate hyper-edges.



This package is a simple implementation of hypergraphs built around the incidence matrix – a binary matrix in which the rows correspond to the hyper-edges, the columns to vertices, and a 1 in position (i,j) indicates that the vertex j is in the ith hyper-edge. There is currently no support for directed or weighted hypergraphs.

Various methods of manipulating hypergraphs, such as adding and removing edges and vertices are implemented, and for small hypergraphs the **igraph** package plot routine is used to plot the hypergraph and its hyper-edges. For hypergraphs with more than a few dozen vertices, it is recommended that the `plot` function be called with `mark.groups=NULL`. See `igraph.plotting` for more information.

There are utilities in this package for removing loops, duplicate hyper-edges, empty hyper-edges, and isolated vertices (ones that are not contained in any hyper-edge). Also, there is a function, `reduce.hypergraph`, which reduces the hypergraph down to its largest hyper-edges – that is, it removes hyper-edges that are subsets of other hyper-edges. It also has other ways to reduce the hypergraph, see the corresponding manual page.

There are also utilities for extracting information from the hypergraph. For example, simple statistics such as the number of vertices, hyper-edges, degrees of vertices, number of nodes per hyper-edge. Also global properties such as whether it is connected, if it has the Helly property or is conformal (see the manual pages for `has.helly` and `is.conformal` for more information on these topics).

Note

Some effort has been taken to avoid masking or redefining functions from the **igraph** package. While this results in awkward function names ("hypergraph" nearly everywhere) it does reduce the chances of hard-to-diagnose errors. I am considering adding aliases that replace "hypergraph" with "hg" or some such, but I'm not sure this is helpful. The two functions that are masked, `is.simple` and `line.graph`, first check whether their argument is an **igraph** graph, and if so calls the corresponding **igraph** function.

Author(s)

David J. Marchette

Maintainer: David J. Marchette <dmarchette@gmail.com>

References

Bretto, Alain, Hypergraph theory, An introduction. Springer, 2013.

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

See Also

`igraph`.

Examples

```
h <- hypergraph_from_edgelist(list(1:2,2:5,3:7,c(1,3,5,7,9)))
hsize(h)
## 4
horder(h)
## 9
```

`as.bipartite` *Hypergraph as a bipartite graph.*

Description

Converts a hypergraph (or graph) into a bipartite graph.

Usage

```
as.bipartite(h)
```

Arguments

h	a hypergraph or a graph.

Details

This converts a hypergraph or a graph into a bipartite graph, by taking the incidence matrix and treating this as the incidence matrix of a bipartite graph. It uses `graph_from_incidence_matrix` to perform the conversion.

Value

an **igraph** bipartite graph.

Note

This works on graphs, resulting in the bipartite graph with edges as one type and vertices as another. This might not be what you want, for example if you think the graph is already bipartite, this will not return the graph, but will rather create a new bipartite graph from the vertices and edges.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

See Also

`graph_from_incidence_matrix`.

Examples

```
h <- hypergraph_from_edgelist(list(letters[1:3],
       letters[c(2,4,7)],
     letters[5:8]))
g <- as.bipartite(h)
```

---

as.hypergraph *Convert between hypergraphs and graphs.*

Description

Convert a hypergraph to a graph or a graph to a hypergraph.

Usage

```
as.graph(h)
hypergraph2graph(h)
as.hypergraph(x,n, ...)
graph2hypergraph(g, method = c("incidence", "adjacency",
                                "neighborhood", "ego",  "spectral"), ...)
```

Arguments

h	a hypergraph.
g	a graph.
method	see Details.
x	a matrix, list or graph. See details.
n	number of vertices if x is missing.
...	arguments passed to various functions. See Details.

Details

For as.graph and hypergraph2graph, create a graph from the incidence matrix using the product of the transpose of the incidence matrix with the incidence matrix. as.graph is an alias of hypergraph2graph. This computes the 2-section of the hypergraph, in the terminology of Bretto.

The function as.hypergraph() returns a hypergraph defined by a graph, matrix (or edgelist). If a matrix is given, it is viewed as the incidence matrix of the hypergraph. If a list is given, it is interpreted as the hyper-edge list. If a graph is given, graph2hypergraph is called with the graph and the arguments passed in .... If x is NULL or missing, and n>0 is given, an empty hypergraph on n nodes is returned. If all else fails, an empty hypergraph on no nodes is returned.

The method variable controls the method used for turning a graph into a hypergraph:

incidence - use the incidence matrix of the graph.

adjacency - treat the adjacency matrix as the incidence matrix.

neighborhood,ego - Use the neighborhoods of the vertices. The arguments to ego are passed in the dotted arguments.

spectral - The spectral embedding is performed, followed by Mclust. The arguments are passed to hypergraph_from_spectral_clustering.

Value

An undirected **igraph** graph object.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Bretto, Alain, Hypergraph theory, An introduction. Springer, 2013.

See Also

as.hypergraph, graph2hypergraph, hypergraph_from_spectral_clustering, ego, Mclust,

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,1:2,c(2,3,5),c(3,5:7)))
g <- as.graph(h)
```

---

ase *Adjacency spectral embedding.*

Description

Using either adjacency or Laplacian spectral embedding, embed a graph into a lower dimensional space.

Usage

```
ase(g, verbose = FALSE, adjust.diag = FALSE, laplacian = FALSE,
    normalize = FALSE, scale.by.values = FALSE, vectors = "u", d = 2)
lse(g,...)
hypergraph.spectrum(h, k=3)
```

Arguments

g, h	A graph (g) or hypergraph (h).
verbose	logical. Control output to terminal.
adjust.diag	logical. For adjacency embedding, whether to add degree/(n-1) to the diagonal of the adjacency matrix.
laplacian	logical. Use the Laplacian rather than the adjacency matrix.
normalize	logical. Whether to normalize by D^1/2.
scale.by.values	logical. Whether to scale the eigen or singular vectors by the square root of the eigen or singular values.
vectors	character. "u", "v" or "uv" indicating which vectors to provide for the embedding.
d, k	dimension of the embedding.
...	arguments passed to ase.

Details

The ase is for graphs, and has the most control over the embedding, as indicated by the arguments. hypergraph.spectrum computes the svd of the incidence matrix for the hypergraph h. lse is Laplacian spectral embedding, and is just a call to ase with laplacian=TRUE and adjust.diag=FALSE. For small hypergraphs (order or size < 3) the base svd function is used and k is ignored.

Value

ase returns a matrix of points, with rows corresponding to vertices and columns to the embedding. There will be either d, or 2*d columns, depending on the value of the variable vectors. For "u" or "v" the dimension is d, for "uv" the dimension is 2*d. hypergraph.spectrum returns the singular value decomposition using the top k singular vectors and values.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Congyuan Yang, Carey E. Priebe, Youngser Park, David J. Marchette, "Simultaneous Dimensionality and Complexity Model Selection for Spectral Graph Clustering," Journal of Computational and Graphical Statistics, accepted for publication, 2020. arXiv:1904.02926

A. Athreya, V. Lyzinski, D. J. Marchette, C. E. Priebe, D. L. Sussman, and M. Tang, "A limit theorem for scaled eigenvectors of random dot product graphs," Sankhya, vol. 78-A, no. 1, pp 1-18, February 2016.

See Also

svds, eigs.

Examples

```
g <- sample_gnp(10,.1)
ase(g)
```

`clique_hypergraph` *Clique Hypergraph*

Description

Construct a clique hypergraph from a graph.

Usage

```
clique_hypergraph(g)
```

Arguments

g	a graph.

Details

A clique hypergraph is one whose hyper-edge correspond to the maximal cliques of a given graph.

Value

a hypergraph.

Warning

The calculation of the maximal cliques of a graph can take a long time, and dense graphs may have many maximal cliques, so use this funciton with care.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

See Also

max_cliques, ~~~

Examples

```
   g <- graph_from_literal(1-2-3-1,3-4-5-3)
h <- clique_hypergraph(g)
```

`cluster_spectral` *Spectral Graph Clustering*

Description

Use spectral embedding to embed a graph into a lower dimension, then cluster the points using model based clustering. This results in a clustering of the vertices.

Usage

```
cluster_spectral(g, verbose = FALSE, adjust.diag = FALSE, laplacian = FALSE,
   normalize = FALSE, scale.by.values = FALSE, vectors = "u", d = 12, ...)
```

Arguments

`g`	a graph.
`verbose`	logical. Whether to print to the screen as it goes.
`adjust.diag`	logical. Whether to set the diagonal of the adjacency matrix to `degree/(n-1)`.
`laplacian`	logical. Whether to use the Laplacian rather than the adjacency matrix.
`normalize`	logical. Whether to normalize the matrix by `D^1/2`.
`scale.by.values`	Whether to scale the embedding vectors by the eigen vectors.
`vectors`	character. "u" or "v" or "uv". The latter is only appropriate for directed graphs.
`d`	embedding dimension.
`...`	arguments passed to `Mclust`.

Details

This first embeds the vertices into a d-dimensional space, using the adjacency matrix or the Laplacian. See `ase` for more information. It then applies `Mclust` to the resultant points to cluster.

Value

An object of class "Mclust".

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Fraley C. and Raftery A. E. (2002) Model-based clustering, discriminant analysis and density estimation, _Journal of the American Statistical Association_, 97/458, pp. 611-631.

See Also

ase.

Examples

```
P <- rbind(c(.2,.05),c(.05,.1))
ns <- rep(50,2)
set.seed(451)
g <- sample_sbm(sum(ns),P,ns)
cluster_spectral(g)
```

---

dual *Dual hypergraph.*

Description

Construct the dual hypergraph of a hypergraph.

Usage

```
dual_hypergraph(h)
```

Arguments

h	a hypergraph.

Details

The dual hypergraph is a hypergraph whose nodes are the original hyper-edges, with hyper-edges indicating the original incidence. Essentially, the incidence matrix of the dual hypergraph is the transpose of the original incidence matrix.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Tyshkevich, R.I. and Zverovich, Vadim E, Line hypergraphs,Discrete Mathematics, 161, 265–283,1996.

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,2:5,4:6,c(1,3,7)))
k <- dual_hypergraph(h)
```

epsilon_hypergraph *Epsilon-Ball Hypergraph*

Description

Build a hypergraph by constructing hyperedges from balls around a set of points.

Usage

```
epsilon_hypergraph(x, epsilon, method = "Euclidean", reduce=FALSE,
   as.graph=FALSE)
```

Arguments

x	a matrix of points.
epsilon	radius of the balls. May be a vector.
method	passed to dist to define the distance function.
reduce	logical. Whether to reduce the hypergraph by removing reduntant hyper-edges.
as.graph	logical. Whether to return a graph instead of a hypergraph.

Details

Each point of x corresponds to a vertex in the hypergraph. For each point, a ball of radius epsilon is constructed, and all points in the ball form a hyper-edge in the graph. If epsilon is a vector, each ball may have a different radius, and if the length of epsilon is less than the number of points, they are repeated.

If reduce=TRUE reduntant hyper-edges (those contained in other hyper-edges) are removed. If as.graph==TRUE, reduce is ignored an a graph is returned instead of a hypergraph.

Value

a hypergraph or graph.

Note

Because of symmetry (a is in the ball centered at b if and only if b is in the ball centered at a), the incidence matrix of an epsilon hypergraph is square and symmetric. It can thus be interpreted as an adjacency matrix, and it is this graph that is returned if as.graph==TRUE.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

See Also

knn_hypergraph, sample_geom_hypergraph, dist.

Examples

```
   set.seed(565)
x <- matrix(rnorm(100),ncol=2)
h <- epsilon_hypergraph(x,epsilon=.25)

   plot(h)
plot(h,layout=x)

epsilons <- runif(nrow(x),0,.5)
k <- epsilon_hypergraph(x,epsilon=epsilons)

   plot(k)
plot(k,layout=x)
```

`equivalent.hypergraphs` *Equivalent Hypergraphs*

Description

Test whether two hypergraphs are equivalent. This is not an isomorphism test, merely a test that the incidence matrices are "the same" in the vertex/edge order in which they are resented.

Usage

```
equivalent.hypergraphs(h1, h2, vertex.names = FALSE, edge.names = FALSE,
strip.names=FALSE,
   method = c("any", "exact", "binary"))
as.binary.hypergraph(h)
```

Arguments

h, h1, h2	hypergraphs.
vertex.names	logical. Whether to ensure the vertex names are all the same.
edge.names	logical. Whether to ensure the hyper-edge names are all the same.
strip.names	logical. Whether to strip the row/column names from the incidence matrices (after ordering them) prior to the equivalence check.
method	see Details.

Details

If either vertex.names or edge.names is TRUE, they are checked for equality, and the incidence matrices are reordered accordingly. The method "exact" checks for the matrices being exactly equal, while "binary" converts all non-zero entries to 1 before the check. The former is for future versions in the event that weighted or directed hypergraphs are implemented. If strip.names is TRUE, the row/column names are stripped from the matrices. If either vertex.names or edge.names is TRUE,

the matrices are first ordered according to the rows/columns as appropriate. Note that "binary" will always be TRUE if "exact" is TRUE, and that if "binary" is FALSE, then so will "exact" be.

The method "any" calls the code with `strip.names=TRUE` and all combinations of `vertex.names` and `edge.names` for the binary method, and returns the logical OR of these. Essentially, this tests that the matrices are "the same" under any reasonable interpretation (without checking for equivalence under any reordering except for lexicographic ordering of the row/column names).

Value

a logical in the case of `equivalent.hypergraphs`, a hypergraph in the case of `as.binary.hypergraph`.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
h1 <- hypergraph_from_edgelist(list(1:4,2:7,c(1,3,5,7),c(2,4,6)))
h2 <- hypergraph_from_edgelist(list(letters[1:4],letters[2:7],
       letters[c(1,3,5,7)],letters[c(2,4,6)]))
equivalent.hypergraphs(h1,h2)  ## TRUE
equivalent.hypergraphs(h1,h2,vertex.names=TRUE)  ## FALSE
```

H2 *Two sections of a hypergraph.*

Description

Two section of a hypergraph.

Usage

```
H2(h)
```

Arguments

h	a hypergraph.

Details

The 2-section of a hypergraph is the graph with vertices corresponding to hyper-edges, and edges corresponding to whether the hyper-edges intersect.

Value

a graph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

Examples

```
h <- hypergraph_from_edgelist(list(c(1,2,5),c(2,3,5),c(3,4),c(4,5)))
g <- H2(h)
## see Figure 7.11 of the reference.
```

---

has.helly *Helly Property*

Description

Check whether a hypergraph has the Helly property.

Usage

```
has.helly(h, strong=FALSE)
is.helly(h)
```

Arguments

h	a hypergraph.
strong	logical.

Details

An intersecting family is a collection of hyper-edges such that the intersection of any pair of hyper-edges in the family is non-empty. A hypergraph has the Helly property if each intersecting family has a non-empty intersection – there is at least one vertex in every hyper-edge. This is an implementation of the algorithm on page 32 of Bretto. The argument `strong` indicates whether the test should be for the strong Helly property or not. A hypergraph has the strong Helly property if every partial induced sub-hypergraph has the Helly property.

The function `is.helly` is an alias for a check for the non-strong Helly property.

Value

a logical.

Note

Have not yet implemented the strong Helly property algorithm.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Bretto, Alain, Hypergraph theory, An introduction. Springer, 2013.

Examples

```
## Example from Bretto
h <- hypergraph_from_edgelist(list(1:5,
          c(2,4,6,7),
 c(4:6,8,9),
 9:10))
   has.helly(h)
```

---

has.isolates *Test for loops, isolates and empty hyper-edges.*

Description

Tools to determine whether a hypergraph has degenerate elements such as loops (hyper-edges with a single vertex) isolated vertices (ones which appear in no hyper-edges) and empty hyper-edges.

Usage

```
has.isolates(h)
has.loops(h)
has.empty.hyperedges(h)
```

Arguments

h	a hypergraph.

Value

returns a logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

See Also

remove.isolates, remove.loops, remove.empty.hyperedges. reduce.hypergraph.

Examples

```
   h <- hypergraph_from_edgelist(list(1:3,2:8,9))
has.loops(h) ## TRUE
has.isolates(h) ## FALSE
k <- hypergraph.add.vertices(h,10)
has.isolates(k) ## TRUE
```

---

hdegree *Degrees of a hypergraph.*

Description

The degree of a vertex in a hypergraph is the number of hyper-edges containing the vertex.

Usage

```
hdegree(h)
plotDegreeDistribution(h, xlab="Degree",
ylab="Density",
add.line=FALSE,
lty=2,lwd=1,line.col=1,
...)
```

Arguments

h	a hypergraph. For plotDegreeDistribution it could also be a graph.
xlab, ylab	axis labels.
add.line	logical. Whether to add a regression line to the plot.
lty, lwd, line.col	plotting controls for the line.
...	arguments passed to plot.

Details

Returns a vector of the number of (hyper-)edges containing each vertex. The plot.hdegree.distributions plots the distribution of degrees on a log-log scale, optionally adding a regression line.

Value

a vector of degrees.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

degree, degree.distribution.

Examples

```
   set.seed(452)
h <- sample_gnp_hypergraph(100,p=.1)
hdegree(h)
```

---

horder *The number of vertices, edges and statistics of the hypergraph.*

Description

This returns the number of vertices and hyper-edges, and similar statistics, for a hypergraph.

Usage

```
hnames(h)
horder(h)
hsize(h)
edge_orders(h)
```

Arguments

h	a hypergraph.

Value

a named vector of vertics, or the names of the vertices. Order refers to the number of vertices, size to the number of hyper-edges. The edge_orders function returns the number of vertices in each of the hyper-edges. In a simple graph, this would always be 2.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

gorder, gsize, hrank.

Examples

```
   h <- hypergraph_from_edgelist(list(3:7,8:12,c(1,3,9)))
horder(h)
hsize(h)
hnames(h)
```

hrank *Rank of a hypergraph.*

Description

Return the rank and corank of a hypergraph. The rank is the maximum cardinality of a hyperedge, the corank (sometimes called the anti-rank) is the minimum.

Usage

```
hrank(h)
```

Arguments

h	a hypergraph.

Value

a number.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Bretto, Alain, Hypergraph theory, An introduction. Springer, 2013.

Examples

```
h <- hypergraph_from_edgelist(list(1:5,4:7,c(1,6)))
hrank(h)   # 5
hcorank(h) # 2
```

hypergraph.add.edges *Add edges or vertices to a hypergraph.*

Description

Adds hyper-edges or vertices to a hypergraph.

Usage

```
hypergraph.add.edges(h, edges, verbose = FALSE)
add.hyperedges(h, edges, verbose = FALSE)
hypergraph.add.vertices(h, nv, names)
```

Arguments

h	A hypergraph.
edges	A list of edges to be added.
nv	Number of vertices to add.
names	Optional vector of names of the vertices.
verbose	logical. Whether to warn if new vertices are created.

Details

The edges can be indices or edge names. This is different than the graph call – see the **igraph** package help for that. If edges is NULL, or missing, `hypergraph.add.edges` adds a single empty hyper-edge to the hypergraph. `add.hyperedges` is an alias for `hypergraph.add.edges`.

Value

Returns a hypergraph (or graph) as appropriate.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

add_vertices, add_edges

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,1:2,c(2,3,5),c(3,5:7)))
h1 <- hypergraph.add.vertices(h,1,"8")
h2 <- hypergraph.add.edges(h,list(c(1,5,8),7:9))
```

---

hypergraph.as.edgelist

Convert between hypergraphs and graphs.

Description

Convert a hypergraph to a graph or a graph, matrix or list to a hypergraph.

Usage

```
hyper_edges(h)
hypergraph_as_edgelist(h)
```

Arguments

h	a hypergraph.

Details

The function hypergraph_as_edgelist is just an alias for hyper_edges.

Value

A list of the hyperedges.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

as_edgelist

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,1:2,c(2,3,5),c(3,5:7)))
hypergraph_as_edgelist(h)
```

---

hypergraph.complement *The complement of a hypergraph.*

Description

The complement of a hypergraph is a hypergraph consisting of the hyper-edges that are not found in the original hypergraph.

Usage

```
hypergraph.complement(h)
```

Arguments

h	a hypergraph.

Details

The incidence matrix of the complement of h has a 0 in those places the original matrix had a 1, and a 1 in those places the original matrix had a 0.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- hypergraph_from_edgelist(list(c(2,3),c(1,4)))
hypergraph.complement(h)
```

`hypergraph.delete.edges` *Delete edges or vertices of a hypergraph.*

Description

Remove edges or vertices from a hypergraph.

Usage

```
hypergraph.delete.edges(h, edges)
delete.hyperedges(h, edges)
hypergraph.delete.vertices(h, v)
```

Arguments

h	a hypergraph.
edges, v	A vector of edges or vertices (indices) to remove.

Details

`delete.hyperedges` is an alias for `hypergraph.delete.edges`.

Value

a hypergraph

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

`delete.vertices`, `delete.edges`.

Examples

```
   h <- hypergraph_from_edgelist(list(1:3,3:7,7:8))
hypergraph.delete.vertices(h,3)
hypergraph.delete.vertices(h,7)
hypergraph.delete.edges(h,2)
```

`hypergraph.entropy` *Hypergraph Entropy*

Description

The hypergraph entropy, which is a sum of the suitably scaled eigenvalues of the hypergraph Laplacian.

Usage

```
hypergraph.entropy(h)
```

Arguments

h	a hypergraph.

Details

Bretto, page 9, defines hypergraph entropy as follows. Let L'(h) be the Laplacian of h divided by the sum of its diagonal. Then the $|V| - 1$ eigenvalues sum to 1, and the entropy is defined by $-sum(\lambda_i \log_2 \lambda_i)$.

Value

a number.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Bretto, Alain, Hypergraph theory, An introduction. Springer, 2013.

See Also

hypergraph_laplacian_matrix.

Examples

```
h <- hypergraph_from_edgelist(list(3:4,1:3,c(3,5,7:10),c(4,6),c(3,5,8)))
hypergraph.entropy(h)
## 2.802822
```

hypergraph.is.connected *Is the hypergraph connected?*

Description

Uses the **igraph** is.connected function to determine if a hypergraph is connected.

Usage

```
hypergraph.is.connected(h)
```

Arguments

h	a hypergraph.

Details

First the hypergraph is converted to a graph. Then the resulting graph is passed to the **igraph** `is.connected` function.

Value

a logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

`is.connected`.

Examples

```
   hypergraph.is.connected(hypergraph_from_edgelist(list(1:4,3:5)))
## TRUE
   hypergraph.is.connected(hypergraph_from_edgelist(list(1:4,5:7)))
## FALSE
```

`hypergraph.union` *Unions and intersections of hypergraphs.*

Description

Given two hypergraphs, compute their union or intersection.

Usage

```
hypergraph.union(h1, h2, reduce = TRUE)
hypergraph.disjoint.union(h1, h2)
hypergraph.intersection(h1, h2, strict = FALSE)
```

Arguments

`h1, h2`	hypergraphs.
`reduce`	logical. Whether to reduce the resultant hypergraph by removing edges that are subsets of other edges.
`strict`	logical. See details.

Details

The disjoint union of two hypergraphs is a hypergraph on the disjoint union of the vertices. The vertices are renamed, if necessary, so that those in the first hypergraph are distinct from those in the second. All edges that occur in either hypergraph, with the vertices renamed, are retained. The (non-disjoint) union treats vertices with the same name (or if they are unnamed, the same index) as the same vertex, and produces the hypergraph containing all edges that are in either hypergraph. If `reduce` is TRUE, the hypergraph is reduced so that hyper-edges that are subsets of another edge are removed. For the intersection, only those edges that are in one of the hypergraphs are retained. Again, vertices with the same name are consider to be the same, and only these vertices are retained. If `strict` is true, the edges must be exactly the same. Otherwise, an edge in one hypergraph that is a subset of an edge in the other will be retained.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h1 <- hypergraph_from_edgelist(list(1:3,3:5,4:9,2:6))
   h2 <- hypergraph_from_edgelist(list(2:3,3:5,4:9,2:6,c(3,5,10:11)))
hypergraph.disjoint.union(h1,h2)
hypergraph.union(h1,h2)
hypergraph.intersection(h1,h2)
```

hypergraph_as_adjacency_matrix *Adjacency Matrix of a Hypergraph.*

Description

Returns the adjacency matrix, computed from the incidence matrix.

Usage

```
hypergraph_as_adjacency_matrix(h)
hadjacency(h)
```

Arguments

h	a hypergraph.

Details

The adjacency matrix is a weighted adjacency matrix corresponding to code(t(M) diagonal of the matrix set to 0. `hadjacency` is an alias for the longer named function.

Value

a (sparse) matrix.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

`as_adjacency_matrix`, `Matrix`.

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,2:5))
hypergraph_as_adjacency_matrix(h)

# returns:
# 5 x 5 Matrix of class "dgeMatrix"
#   1 2 3 4 5
# 1 0 1 1 1 0
# 2 1 0 2 2 1
# 3 1 2 0 2 1
# 4 1 2 2 0 1
# 5 0 1 1 1 0
```

`hypergraph_from_incidence_matrix` *Hypergraph construction.*

Description

Construct a hypergraph from a collection of hyper-edges.

Usage

```
hypergraph_from_incidence_matrix(incidence_matrix)
hypergraph_from_edgelist(x,v)
hypergraph_from_membership(x)
hypergraph_from_fuzzy_clustering(z,threshold)
hypergraph_from_spectral_clustering(g,m,fuzzy=FALSE,threshold,...)
```

Arguments

`incidence_matrix`	an sxn binary matrix corresponding to the s hyper-edges on n vertices.
`x`	a list of hyper-edges, or a vector corresponding to which hyper-edge each node is in.
`z`	a matrix of probabilities that is nxc where c is the number of clusters.
`v`	optional vector of node names.
`g`	a graph. Only used if `m` is missing.
`m`	a communities object. See communities in the `igraph` package.
`fuzzy`	logical.
`threshold`	threshold on the probabilities if `fuzzy` is true. If not given, it defaults to the inverse of the number of communities.
`...`	arguments passed to `cluster_spectral` if `g` is given and `m` is given.

Details

An edgelist is a list of hyper-edges. An incidence matrix is a binary matrix that is `hsize(g)xhorder(h)`. A membership vector is the vector of node membership returned from a community detection or clustering algorithm. A hypergraph constructed from a membership vector has a disconnected component for each hyper-edge.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

cluster_spectral, communities.

Examples

```
h <- hypergraph_from_edgelist(list(1:4,3:6))
```

---

hypergraph_from_literal *Hypergraph from literal.*

Description

Similar to the **igraph** function, except that in this case the literals just indicate the hyper-edges.

Usage

```
hypergraph_from_literal(...)
```

Arguments

...	see details.

Details

This takes a collection of hyperedge descriptions, such as a-3-C-9, indicating the hyper-edge containing a,C,3 and 9. It returns the associated hypergraph. This is different from the graph version; in essence, this would be the same as constraining the graph version to require all the argumenst to be pairs. There is no way to produce chains, as in the graph version.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

graph_from_literal.

Examples

```
h1 <- hypergraph_from_literal(1-2-3,3-a-b,c-1-4)

plot(h1)
```

`hypergraph_laplacian_matrix` *Laplacian Matrix*

Description

The Laplacian of a hypergraph is D-A, where A is the (weighted) adjacency matrix, and D is the row-sums.

Usage

```
hypergraph_laplacian_matrix(h,normalize=FALSE)
```

Arguments

`h`	a hypergraph.
`normalize`	logical. Whether to normalize the Laplacian matrix.

Details

The Laplacian is D-A, where D is the row sums of the adjacency matrix A. If `normalize` is TRUE, then the normalized version is returned.

Value

a (sparse) matrix.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

`laplacian_matrix`.

Examples

```
h <- hypergraph_from_edgelist(list(1:4,2:5,c(3,5,7,8,10),c(2,9),c(2:3,6,10)))
L <- hypergraph_laplacian_matrix(h)
Ln <- hypergraph_laplacian_matrix(h,normalize=TRUE)
```

incidence_matrix *Graph Incidence Matrix.*

Description

Converts a graph to an incidence matrix. Not the bipartite version.

Usage

```
incidence_matrix(g)
hypergraph_as_incidence_matrix(h)
```

Arguments

g	a graph or hypergraph.
h	a hypergraph.

Details

An incidence matrix has gorder(g) columns and gsize(g) rows. incidence_matrix can be called on either a graph or a hypergraph. It calls hypergraph_as_incidence_matrix in the latter case.

Value

A sparse incidence matrix.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
set.seed(2343)
   g <- sample_gnp(10,.1)
h <- hypergraph_from_edgelist(list(1:3,3:4,c(3,5,7)))
```

`induced_hypergraph` *Induced hypergraph.*

Description

Computes the hypergraph induced by a subset of the vertices.

Usage

```
induced_hypergraph(h, v, simplify = TRUE)
```

Arguments

`h`	a hypergraph.
`v`	a vector of vertices.
`simplify`	logical.

Details

First the hypergraph is reduced to only those vertices in `v`. This results in it retaining only those hyper-edges containing any elements of `v`, as well as removing from the resultant hyper-edges any vertices not in `v`. If `simplify` is true, loops are then removed. This function always removes empty hyper-edges, so any hyper-edge which does not contain any elements of `v` is removed.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,3:7,c(1,3,5)))
k <- induced_hypergraph(h,c(1,3,5))
```

`is.conformal` *Conformal Hypergraphs*

Description

Tests whether a hypergraph is conformal.

Usage

```
is.conformal(h)
is.bi.conformal(h)
```

Arguments

h	a hypergraph.

Details

A hypergraph h is conformal if all the maximal cliques of its 2-section are the maximal (by inclusion) edges of h. The test uses a theorem (see the reference, Theorem 7.6.4) that says a hypergraph is conformal if and only if its dual is Helly. A hypergraph is bi-conformal if it and its dual are conformal.

Value

a logical.

Author(s)

David J. Marchete <dmarchette@gmail.com>

References

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

See Also

`is.helly`.

Examples

```
h <- hypergraph_from_edgelist(list(1:4,3:6,5:10))
is.conformal(h)
## TRUE
h <- hypergraph_from_edgelist(list(1:2,2:3,c(1,3)))
is.conformal(h)
## FALSE
```

is.empty.hypergraph *Is the hypergraph empty.*

Description

determines whether the hypergraph contains no hyper-edges.

Usage

```
is.empty.hypergraph(h)
```

Arguments

h	a hypergraph.

Value

a logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
h <- hypergraph_from_edgelist(list(1:3))
is.empty.hypergraph(h)
k <- hypergraph.delete.edges(h,1)
is.empty.hypergraph(k)
```

is.hypergraph *Is an object a hypergraph?*

Description

Check that an object is a hypergraph object.

Usage

```
is.hypergraph(h)
```

Arguments

h	a hypergraph.

Details

This only checks that the object's class contains hypergraph.

Value

A logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,1:2,c(2,3,5),c(3,5:7)))
g <- as.graph(h)
is.hypergraph(h)
is.hypergraph(g)
```

---

is.hypertree *Test for hypertree.*

Description

Test if a hypergraph is a hypertree.

Usage

```
is.hypertree(h, ...)
```

Arguments

h	a hypergraph.
...	arguments passed to the **igraph** is_chordal function.

Details

Uses Corollary 8.1.1 of the reference: a hypergraph is a hypertree if and only if it is Helly and it's line graph is chordal.

Value

a logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

References

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

See Also

is_chordal. line_graph. line.graph, has.helly.

Examples

```
h <- hypergraph_from_edgelist(list(1:3,2:4,4:5,c(2,4:5)))
is.helly(h)
g <- line.graph(h)
is_chordal(g)
is.hypertree(h)
```

---

is.simple *Is a hypergraph simple/linear?*

Description

Tests whether a hypergraph is simple or linear.

Usage

```
is.simple(h)
is.linear(h)
```

Arguments

h	a hypergraph. Can be a graph for is.simple, in which case the **igraph** version is called.

Details

A hypergraph is simple if all it's edges are distinct, non-empty, and if edge i is contained in edge j, then i=j. A hypergraph is linear if it is simple and the intersection of any two hyper-edges has at most one element.

Value

a logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

References

Akram, M., & Luqman, A. (2020). Fuzzy hypergraphs and related extensions. Springer Singapore.

Examples

```
h <- hypergraph_from_edgelist(list(1:4,4:7,c(6,8:10),10:14))
is.linear(h)  ## TRUE
is.simple(h)  ## TRUE
```

`is.star` *Is a hypergraph a star?*

Description

Tests whether a hypergraph is a star. Finds the minimal intersection set of the hyper-edges.

Usage

```
is.star(h,type=c("weak","strong"))
intersection_set(h)
```

Arguments

`h`	a hypergraph.
`type`	see Details.

Details

A (weak) star hypergraph is one in which the intersection of all the hyper-edges is non-empty.

An intersection set is a set of vertices that is contained in every edge. The argument `type="strong"` for `is.star` indicates that the only vertices which are common between any pair of vertices are contained in the intersection set (or "hub") of the star hypergraph. So a hypergraph that is strongly star is weakly star, but not vice versa.

Value

a logical, for `is.star`. A set of vertices (or NULL) that are contained in every hyper-edge. This would be the "hub" of the star.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

References

Akram, M., & Luqman, A. (2020). Fuzzy hypergraphs and related extensions. Springer Singapore.

Examples

```
h <- hypergraph_from_edgelist(list(c(1:4,16),
  c(4:7,16),
  c(4,8:10,16),
  c(4,10:16),
  c(4,16)))
is.star(h)  ## TRUE
is.star(h,type='strong')  ## FALSE
intersection_set(h)     ## 4, 16
```

`is.tree` *Test if a graph is a tree or a forest.*

Description

Test if a graph is a tree or a forest.

Usage

```
is.tree(g)
is.forest(g,strict=FALSE)
```

Arguments

g	a graph.
strict	logical.

Details

test whether an undirected graph g is a tree (connected, acyclic) or a forest (disjoing union of trees). The flag `strict` enforces the strict rule that a forest must contain more than a single tree. The default is to allow for single-tree forests, which is the convention.

Value

a logical.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   g <- make_tree(10)
is.tree(g)
is.forest(g)
is.forest(g,strict=TRUE)
```

kCores *K-Cores*

Description

Find all the k-cores in a hypergraph.

Usage

```
kCores(h)
```

Arguments

h	a hypergraph.

Details

A k-core in a hypergraph is a maximal subhypergraph where (a) no hyperedge is contained in another, and (b) each node is adjacent to at least k hyperedges in the subgraph.

The implementation is based on the algorithm by E. Ramadan, A. Tarafdar, A. Pothen, 2004.

The code is a direct copy of the code from the BioConductor package hypergraph, modified to work with the data structures used in this package.

Value

A vector of core numbers for each vertex.

Author(s)

David J. Marchette <dmarchette@gmail.com>

The implementation in the hypergraph package from which this function was taken was written by:

Li Long <li.long@isb-sib.ch>.

References

A hypergraph model for the yeast protein complex network, Ramadan, E. Tarafdar, A. Pothen, A., Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International.

See also the BioConductor hypergraph package:

Seth Falcon and Robert Gentleman (2020). hypergraph: A package providing hypergraph data structures. R package version 1.62.0

Examples

```
  ## example from the hypergraph version
  edges <- list(c("A", "C"),
        c("B", "C"),
  c("C", "E"),
  c("C", "F"),
  c("E", "D"),
  c("E", "F"),
  c("D", "G"),
  c("D", "H"),
  c("D", "J"),
  c("H", "G"),
  c("H", "J"),
  c("G", "J"),
  c("J", "M"),
  c("J", "K"),
  c("M", "K"),
  c("M", "O"),
  c("M", "N"),
  c("K", "N"),
  c("K", "F"),
  c("K", "I"),
  c("K", "L"),
  c("F", "I"),
  c("I", "L"),
  c("F", "L"),
  c("P", "Q"),
  c("Q", "R"),
  c("Q", "S"),
  c("R", "T"),
  c("S", "T"))
h <- hypergraph_from_edgelist(edges,v=union(unlist(edges),"U"))
kc <- kCores(h)

kCores(h)
```

---

knn_hypergraph *K-Nearest Neighbor Hypergraph.*

Description

A hypergraph is constructed from data in which each hyper-edge corresponds to a vertex and its k-nearest neighbors.

Usage

```
knn_hypergraph(x, k = 1, method = "Euclidean", reduce=FALSE,
   as.graph=FALSE)
```

Arguments

x	a matrix of data points.
k	the number of neighbors. May be a vector.
method	distance type passed to dist.
reduce	logical. Whether to remove redundant hyper-edges.
as.graph	logical. Whether to return a graph instead of a hypergraph.

Details

Each vertex is in one-to-one correspondence with the points (rows) of x. For each vertex, the k-closest vertices and itself form a hyper-edge.

If reduce=TRUE reduntant hyper-edges (those contained in other hyper-edges) are removed. If as.graph=TRUE, reduce is ignored and the incidence matrix is treated as an adjacency matrix, returning a (directed) **igraph** graph.

Value

a hypergraph or graph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

epsilon_hypergraph, dist.

Examples

```
   set.seed(565)
x <- matrix(rnorm(100),ncol=2)
k <- knn_hypergraph(x,k=4)

   plot(k)
plot(k,layout=x)
```

---

line.graph *Line Graph*

Description

Construct the line graph of a hypergraph.

Usage

```
line.graph(h)
```

Arguments

h	a hypergraph.

Details

The line graph of a hypergraph is essentially the same concept as the line graph of a graph: it is the graph whose vertices correspond to the hyper-edges, with an edge between two vertices if their corresponding hyper-edges intersect.

Value

a graph.

Note

If h is an **igraph** graph, the function `igraph::line_graph` will be called. `line.graph` is the only function that masks an **igraph** function, but I am assuming that the '_' version is preferred by **igraph**.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

See Also

`line_graph`.

Examples

```
h <- hypergraph_from_edgelist(list(1:3,2:4,4:5,c(2,4:5)))
g <- line.graph(h)
```

---

make_empty_hypergraph *Empty hypergraph.*

Description

Create an empty hypergraph.

Usage

```
make_empty_hypergraph(n)
```

Arguments

n	a non-negative integer.

Details

Creates an empty hypergraph (no hyper-edges) on n nodes.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- make_empty_hypergraph(4)
horder(h)
hsize(h)
h <- make_empty_hypergraph(0)
horder(h)
hsize(h)
```

---

pendant *Pendant Vertices*

Description

Determine the set of pendant vertices.

Usage

```
pendant(h)
```

Arguments

h	a hypergraph.

Details

A pendant vertex is one whose set of hyper-edges is a subset of the hyper-edges of another vertex. That is, if v is in hyper-edges 1 and 2, and w is in hyper-edges 1, 2 and 5, then v is pendant to w, and w is called a twin of v.

Value

a list containing:

vertices	a vector of the pendant vertices
twins	a list, each element of which is the set of twins of the corresponding pendant vertex

Author(s)

David J. Marchette <dmarchette@gmail.com>.

References

Voloshin, Vitaly I. Introduction to graph and hypergraph theory. Nova Science Publ., 2009.

Examples

```
h <- hypergraph_from_edgelist(list(1:4,2:5,c(4,6),c(7),c(3:5,8)))
pendant(h)
```

---

plot.hypergraph *Plot a hypergraph.*

Description

Plot a hypergraph using the **igraph** plot function.

Usage

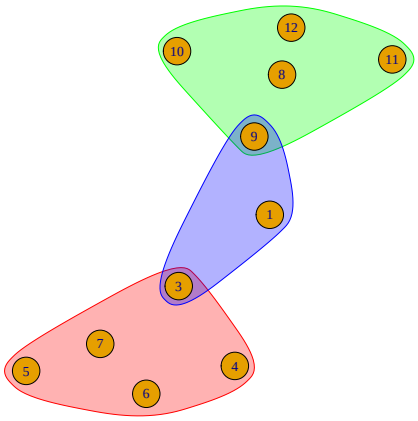
```
## S3 method for class 'hypergraph'
plot(x, edge.color = NA,
 mark.groups=hypergraph_as_edgelist(h),
    layout,...)
```

Arguments

x	a hypergraph.
edge.color	color for the edges.
layout	optional layout for the plot. If the hypergraph has a layout attribute, this will be used, unless layout is given. if it does not have a layout attribute and the layout is not provided, it uses the code from **igraph** to choose a layout.
mark.groups	the groups correspond to the hyper-edges. Set this to NULL if you do not want the hyper-edge polygons to plot.
...	optional arguments passed to plot.

Details

Plots the hypergraph, using the **igraph** plotting function applied to a graph converted from the hypergraph. For the example below, the plot will look something like:



Value

the layout is returned invisibly.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

as.graph, plot.igraph, igraph.plotting.

Examples

```
h <- hypergraph_from_edgelist(list(3:7,8:12,c(1,3,9)))

plot(h)
```

`print.hypergraph` *Print a hypergraph to the console.*

Description

Print method for hypergraphs.

Usage

```
## S3 method for class 'hypergraph'
print(x, ...)
```

Arguments

x	a hypergraph.
...	ignored arguments.

Value

No return value, called for side effects only – prints to console.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- hypergraph_from_edgelist(list(3:7,8:12,c(1,3,9)))
print(h)
```

---

reduce.hypergraph *Remove redundant hyperedges and isolated vertices.*

Description

Reduce a hypergraph by removing redundant edges, loops, or isolated vertices.

Usage

```
reduce.hypergraph(h,method="inclusion")
simplify.hypergraph(h)
remove.isolates(h)
remove.loops(h)
remove.empty.hyperedges(h)
remove.loops(h)
```

Arguments

h	a hypergraph
method	character. See details.

Details

reduce.hypergraph removes redundant edges and/or reduces the hyper-edges: if the method is "inclusion" this removes hyper-edges that are contained in other hyper-edges; if the method is "intersection", it replaces the hypergraph with a new hypergraph whose edges are intersections of the original hypergraph's hyper-edges – each pair of hyper-edges e1 and e2 with a non-empty intersection result in a hyper-edge corresponding to that intersection in the new hypergraph; if the method is "union" it removes edges that are contained in the union of the other edges, reducing to a hypegraph in which every hyper-edge contains at least one vertex which is contained in no other hyper-edge.

Isolates are vertices that are not in any hyper-edges, and loops are hyper-edges containing a single vertex. reduce.hypergraph will remove edges which contain no vertices, since the null set is a subset of any hyper-edge and hence is in the union of all other hyper-edges, so for either method such edges would be removed. simplify.hypergraph removes loops, isolates, and empty hyper-edges from a hypergraph. Isolates are removed after removing loops. This may reduce the order of the hypergraph, unlike the **igraph** simplify command, which only removes edges.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- hypergraph_from_edgelist(list(1:4,1:3,2:4,2:6))
reduce.hypergraph(h)
```

---

remove.redundant.vertices

Remove redundant vertices.

Description

Remove vertices whose removal does not disconnect the hypergraph.

Usage

```
remove.redundant.vertices(H, check.empty = TRUE)
```

Arguments

H	a hypergraph.
check.empty	logical. See details.

Details

This function was created with a binary term document hypergraph in mind. The idea is to remove words that appear in large documents, in order of decreasing degree, so long as the removal does not disconnect the graph. The argument `check.empty` is to ensure that removing a word does not result in an empty document. If this is set to FALSE, the resultant hypergraph may be much smaller than expected.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
 edges <- list(letters[c(1:4,9)],
letters[3:9],
letters[c(3,6:9)],
letters[c(3,5:9)],
letters[2:9])
 h1 <- hypergraph_from_edgelist(edges)
 h <- remove.redundant.vertices(h1)
 # removed c, f, g, i

 h2 <- hypergraph_from_edgelist(list(letters[1:3],
 letters[3:5]))

 h <- remove.redundant.vertices(h2)
 ## h == h2
```

---

reorder_vertices *Reorder the vertices of a hypergraph.*

Description

Return a hypergraph in which the vertices have been reordered so that they are in the given order as columns of the incidence matrix.

Usage

```
reorder_vertices(h,ord,decreasing=FALSE)
```

Arguments

`h`	a hypergraph.
`ord`	an ordering of the vertices.
`decreasing`	if `ord` is not given, the order function is called on the names of the vertices as controlled by the `decreasing` variable.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

order

Examples

```
   h <- hypergraph_from_edgelist(list(3:7,8:12,c(1,3,9)))
k <- reorder_vertices(h)
hnames(h)
hnames(k)

   h <- hypergraph_from_edgelist(list(letters[3:7],letters[8:12],
                                      LETTERS[c(1,3,9)]))
k <- reorder_vertices(h)
hnames(h)
hnames(k)
```

---

sample_geom_hypergraph

Construct a hypergraph from a random collection of points.

Description

A hypergraph defined by the relationships amongst a set of points.

Usage

```
sample_geom_hypergraph(n, m, d = 2, X, Y, radius,method = "Euclidean",
thresh.method="leq", uniformly = FALSE)
plot_geom_hypergraph(h, pch = 20, cex = 3, col = "gray",
plotY = TRUE, plot.circles = plotY,
full.circles=TRUE,
lty = 2, lcol = "black", ...)
```

Arguments

n	The number of points to generate (ignored if X is provided). This corresponds to the number of vertices in the hypergraph.
m	The number of nodes in the hypergraph (ignored if Y is provided). This corresponds to the number of hyper-edges in the hypergraph.

d	dimension of the points (ignored if both X and Y are provided).
radius, uniformly	see Description.
X	see Description.
Y	see Description.
method	method passed to dist.
thresh.method	if this is 'leq' then hyper-edges are defined by whether the Y points are a distance less than or equal to the radius. If 'geq', then it is determined by greater than. Any other value will default to 'leq'.
h	a hypergraph generated by sample_geom_hypergraph.
plotY	logical. Whether to plot the Y variables defining the hypergaph.
pch, cex, col	parameters controling the plotting of Y.
plot.circles, full.circles	logical. Whether to plot the circles defining the hyper-edges. If full.circles is TRUE, the plot limits are set so that the complete circles are plotted. Otherwise they may be clipped.
lty, lcol	parameters controling the plotting of the circles.
...	parameters passed to plot.

Details

If either X or Y is missing, it is generated as a set of d-dimensional points in the unit cube, n points in X, m points in Y. If X is given, then n is ignored. Similary with Y and m. If both are given, then d is ignored. There is no checking that the provided X and/or Y matrices conform to the n,m,d values given in the call.

The inter-point distance matrix is computed using dist as proxy::dist(Y,X,method=method). If radius is not provided, it is chosen uniformly at random from the unique values of the distance matrix if uniformly is FALSE, and uniformly from between the minimum and maximum distance if uniformly is true. As a rule, one should not let this function choose the radius, but the code will do so if you wish. The matrix is then thresholded by the radius, resulting in a binary matrix which is then used as the mxn incidence matrix for the hypergraph.

Value

a hypergraph. Additionally, the defining vectors X and Y and the radius are returned as named values of the hypergraph.

Note

If both X and Y are given, and R is not given, a random value for R is chosen randomly from the unique inter-point distances. If all three of these variables are provided, the hypergraph is not random.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

dist.

Examples

```
set.seed(235)
   h <- sample_geom_hypergraph(100,20,radius=0.2)
set.seed(3519)
Y <- matrix(runif(20),ncol=2)
X <- do.call(rbind,
     lapply(1:nrow(Y),function(i) cbind(rnorm(10,Y[i,1]),rnorm(10,Y[i,2]))))


   h <- sample_geom_hypergraph(X=X,Y=Y,radius=0.2)
```

---

sample_gnp_hypergraph *Erdos-Renyi hypergraphs.*

Description

Sample an Erdos-Renyi p hypergraph.

Usage

```
sample_gnp_hypergraph(n, m, p, lambda)
```

Arguments

n	number of nodes.
m, lambda	controls the number of hyper-edges. If m is not given, the number is drawn from a Poisson(lambda), or, a Poisson(n*p) if lambda is not given.
p	Hyper-edge probability.

Details

This generates an ER hypergraph by using rbinom to generate a random mxn matrix of Bernoulli random variables and treating this matrix as the incidence matrix for the hypergraph. If m is not given, and lambda is, then m is drawn from a Poisson distribution with parameter lambda. If neither is given, the number of hyper-edges is drawn from a Poisson distribution with parameter n*p.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

rbinom, rpois.

Examples

```
set.seed(672)
   h <- sample_gnp_hypergraph(n=100,p=.1)
```

---

sample_k_uniform_hypergraph

Random k-uniform and k-regular hypergraphs.

Description

Randomly generate a hypergraph in which each hyper-edge contains k vertices, or each vertex is incident to k hyper-edges.

Usage

```
sample_k_uniform_hypergraph(n, m, k, prob)
sample_k_regular_hypergraph(n, m, k, prob)
```

Arguments

n	the order of the hypergraph.
m	the size of the hypergraph.
k	the order of each hyper-edge.
prob	a vector of length n containing the probabilities for the vertices. This is passed to sample.

Details

A k-uniform hypergraph is one for which each hyper-edge contains exactly k vertices. A k-regular hypergraph is one for which each vertex has degree k. These are implemented through calls to sample.

Value

a hypergraph.

Note

For both of these functions m, the number of hyper-edges, must be provided. This is unlike the corresponding functions for graphs – all simple graphs are 2-uniform, and there are constraints on the values of k for which a graph can be k-regular, since the graphs are constrained to be 2-regular, i.e. all rows of the incidence matrix must contain 2 ones.

Author(s)

David J. Marchette <dmarchette@gmail.com>.

See Also

sample.

Examples

```
set.seed(77)
h <- sample_k_uniform_hypergraph(20,5,3)
set.seed(73)
k <- sample_k_regular_hypergraph(20,5,3)
```

---

sample_sbm_hypergraph *Sample from a stochastic block model.*

Description

A stochastic block model hypergraph.

Usage

```
sample_sbm_hypergraph(n,P,block.sizes,d,impurity=0,variable.size=FALSE,
   absolute.purity=TRUE)
```

Arguments

n	number of vertices.
P	A kxk probability matrix.
block.sizes	vector of community sizes.
d	size of a hyper-edge. See Details.
impurity	See Details.
variable.size, absolute.purity	logical. See Details.

Details

A stochastic block model is first generated using the function sample_sbm(n,P,block.sizes). The edges are augmented with vertices, resulting in a stochastic block model hypergraph, as discussed below.

The variable d corresponds to the number of vertices per edge. If it is a vector, it is recycled as necessary. If variable.size is TRUE, then d is used as the mean of a Poisson random variable to generate hyper-edge orders, to which 2 is added. So a d of 2 will result in hyper-edge orders with a mean of 4.

For each edge (say, edge k) in the graph, new vertices are added so that the number of vertices in the (now hyper-)edge is d[k].

If `impurity` is 0, then for each edge the vertices are added in proportion to the block sizes, using `sample`, so if one community has many more vertices than the others, it will tend to dominate in the hyper-edges as well. However, it is guaranteed that hyper-edges between two distince communities have at least one vertex from each of those two communities, and hyper-edges within communities are pure in the case of `impurity=0`; no hyper-edge will contain vertices from more than two communities.

If `impurity>0`, then `impurity` of the vertices not in the original stochastic block model hypergraph are replaced by random vertices. If `absolute.purity` is TRUE, these new vertices are sampled from all other classes. Otherwise they are sample from vertices not in the original hyper-edge. If k is 2, only the within community hyper-edges will contain impurities.

Value

a hypergraph.

Author(s)

David J. Marchette <dmarchette@gmail.com>

See Also

sample_gnp_hypergraph, sample_sbm.

Examples

```
P <- rbind(c(0.1,0.01),c(0.01,0.1))
block.sizes <- c(50,50)
set.seed(55)
   h <- sample_sbm_hypergraph(100,P=P,block.sizes=block.sizes,d=4)
range(edge_orders(h))
## should all be 4
set.seed(1233)
k <- sample_sbm_hypergraph(100,P=P,d=2,block.sizes=block.sizes,
             variable.size=TRUE)
mean(edge_orders(k))
## should be approximately 4
set.seed(1235)
Q <- rbind(c(.2,.01,.01),
           c(.01,.1,.05),
  c(.01,.05,.2))
kk <- sample_sbm_hypergraph(300,P=Q,d=6,block.sizes=rep(100,3),
             variable.size=TRUE,impurity=2)
```

`subtree.hypergraph` *Subtree Hypergraph.*

Description

Construct a subtree hypergraph from a graph.

Usage

```
subtree.hypergraph(g, v)
```

Arguments

g	a graph.
v	a list of vertex sets. See details.

Details

A subtree hypergraph is a hypergraph on the vertices of the graph g, each of whose hyper-edges induces a subtree in g. If v is given, each element of the list must contain at least two elements. For each element of v, all paths between it's first element and each of the other elements are computed, and the hyper-edge corresponds to all the vertices in these paths.

Value

a hypergraph.

Note

There are many possible subtree hypergraphs for a given graph. The default is to loop through the vertices in a particular way to generate a hypergraph. The intent is that the use should provide v, rather than using this admittedly arbitrary algorithm.

Author(s)

David J. Marchette <dmarchette@gmail.com>

References

Bretto, Alain, Hypergraph theory, An introduction. Springer, 2013.

Examples

```
g <- make_tree(20,mode='undirected')
```

summary.hypergraph *Print a summary of the hypergraph to the console.*

Description

Summary method for hypergraphs.

Usage

```
## S3 method for class 'hypergraph'
summary(object, ...)
```

Arguments

object	a hypergraph.
...	ignored arguments.

Value

No return value, called for side effects only – prints to console.

Author(s)

David J. Marchette <dmarchette@gmail.com>

Examples

```
   h <- hypergraph_from_edgelist(list(3:7,8:12,c(1,3,9)))
print(h)
summary(h)
```

Index