

Package ‘archeofrag’

March 7, 2025

Type Package

Title Spatial Analysis in Archaeology from Refitting Fragments

Version 1.1.0

Date 2025-03-07

Author Sebastien Plutniak [aut, cre] (<<https://orcid.org/0000-0002-6674-3806>>)

Maintainer Sebastien Plutniak <sebastien.plutniak@posteo.net>

Description

Methods to analyse spatial units in archaeology from the relationships between refitting fragmented objects scattered in these units (e.g. stratigraphic layers). Graphs are used to model archaeological observations. The package is mainly based on the 'igraph' package for graph analysis. Functions can: 1) create, manipulate, and simulate fragmentation graphs, 2) measure the cohesion and admixture of archaeological spatial units, and 3) characterise the topology of a specific set of refitting relationships. Empirical datasets are provided as examples. Documentation about 'archeofrag' is provided by the vignette included in this package, by the accompanying scientific papers: Plutniak (2021, Journal of Archaeological Science, <[doi:10.1016/j.jas.2021.105501](https://doi.org/10.1016/j.jas.2021.105501)>) and Plutniak (2022, Journal of Open Source Software, <[doi:10.21105/joss.04335](https://doi.org/10.21105/joss.04335)>). This package is complemented by a companion GUI application available at <<https://analytics.huma-num.fr/Sebastien.Plutniak/archeofrag/>>.

License GPL-3

Repository CRAN

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports igraph, graphics, stats, grDevices, methods, utils

Suggests RBGL, knitr, covr, rmarkdown, markdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://github.com/sebastien-plutniak/archeofrag>

BugReports <https://github.com/sebastien-plutniak/archeofrag/issues>

NeedsCompilation no

Date/Publication 2025-03-07 16:20:09 UTC

Contents

archeofrag-package	2
BoutDesVergnes	3
Chauzeys	4
CuzoulCave	5
CuzoulSouth	6
FontJuvenal	7
frag.cycles	8
frag.diameters	9
frag.edges.weighting	10
frag.get.layers	12
frag.get.layers.pair	13
frag.get.parameters	14
frag.graph.plot	16
frag.graph.reduce	17
frag.layers.admixture	18
frag.layers.cohesion	19
Frag.object-class	21
frag.observer.failure	22
frag.path.lengths	23
frag.relations.by.layers	24
frag.simul.compare	25
frag.simul.process	26
frag.simul.summarise	28
GrandeRivoire	30
LiangAbu	31
make_crsr_graph	32
make_cr_graph	33
make_frag_object	34
make_sr_graph	35
TaiCave	36
TaiSouth	37
Index	38

archeofrag-package *Archeofrag: Refitting and Spatial Analysis in Archaeology*

Description

Methods to analyse fragmented objects in archeology using refitting relationships between fragments scattered in archeological spatial units (e.g. stratigraphic layers). Graphs and graph theory are used to model archeological observations. The package is mainly based on the 'igraph' package for graph analysis. Functions can: 1) create, manipulate, and simulate fragmentation graphs, 2) measure the cohesion and admixture of archeological spatial units, and 3) characterise the topology of a specific set of refitting relationships. An empirical dataset is also provided as an example.

Author(s)

Sebastien Plutniak Maintainer: Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[igraph](#), [RBGL](#)

BoutDesVergnes

Dataset: Refitting relationships between lithic fragments from the Bout des Vergnes site

Description

Bout des Vergnes is an open-air Palaeolithic site located in France. This dataset describes the refitting 'connection' relationships between lithic fragments found during excavations. Connection relationship refers to physical connection between two fragments that were part of the same object.

The dataset is composed of two tables:

- `bdv.connection`: connection relationships between fragments.
- `bdv.fragments`: information about the fragments.

Usage

```
data(BoutDesVergnes)
```

Format

- `bdv.connection` is a 3931x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `bdv.fragments` is 3666x9 data frame:
 - `id`: fragment unique identifier
 - `layer`: chronocultural unit associated to the fragment was found. Values are numbered in expected chrono-stratigraphic order
 - `x`: x coordinate of the location where the lithic fragment was found (in metre)
 - `y`: y coordinate of the location where the lithic fragment was found (in metre)
 - `z`: z coordinate of the location where the lithic fragment was found (in metre)

References

- Ihuel, E. (dir.), with Baillet M., Barbeyron A., Brenet M., Camus H., Claud E., Mercier N., Michel A., Sellami F. 2020. Le Bout des Vergnes, Bergerac (Dordogne, Nouvelle-Aquitaine), Contournement ouest de Bergerac, RD 709, Rapport final d'opération, Périgueux, Conseil départemental de la Dordogne, Service départemental d'archéologie.

Examples

```
data(BoutDesVergnes)
head(bdv.fragments)
```

Chauzeys

Dataset: Refitting relationships between lithic fragments from the Chauzeys site

Description

Chauzey is an open-air Palaeolithic site located in France. This dataset describes the refitting 'connection' relationships between lithic fragments found during excavations. Connection relationship refers to physical connection between two fragments that were part of the same object.

The dataset is composed of two tables:

- `chauzeys.connection`: connection relationships between fragments.
- `chauzeys.fragments`: information about the fragments.

Usage

```
data(Chauzeys)
```

Format

- `chauzeys.connection` is a 1879x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `chauzeys.fragments` is a 2167x9 data frame:
 - `id`: fragment unique identifier
 - `layer`: chronocultural unit associated to the fragment was found. Values are numbered in expected chrono-stratigraphic order
 - `volume`: estimated volume of the fragment computed as width by thickness by length
 - `width`: width of the fragment (in millimeter)
 - `thickness`: thickness of the fragment (in millimeter)
 - `length`: length of the fragment (in millimeter)
 - `x`: x coordinate of the location where the lithic fragment was found (in metre)
 - `y`: y coordinate of the location where the lithic fragment was found (in metre)
 - `z`: z coordinate of the location where the lithic fragment was found (in metre)

References

- Chadelle J.-P. (dir.), with M. Baillet, A. Barbeyron, M. Brenet, H. Camus, É. Claud, F. Jude, S. Kreuzer, A. Michel, N. Mercier, M. Rabanit, S. Save, F. Sellami, A. Vaughan-Williams. 2021. Chauzeys, Saint-Médard-de-Mussidan (Dordogne, Nouvelle-Aquitaine), Rapport final d'opération archéologique, Périgueux, Conseil départemental de la Dordogne, Service départemental d'archéologie.

Examples

```
data(Chauzeys)
head(chauzeys.fragments)
```

CuzoulCave

Dataset: Refitting relationships between fauna fragments from the Cuzoul de Gramat site

Description

The Cuzoul de Gramat site is a prehistoric site located in France. This dataset describes the refitting 'connection' relationships between bone fragments found during the excavation of Mesolithic layers from the 'Cave' sector of the site. Connection relationship refers to physical connection between two bone fragments that were part of the same object.

The dataset includes two tables:

- `cuzoul.cave.connection`: connection relationships between fragments.
- `cuzoul.cave.fragments`: information about the fragments.

Usage

```
data(CuzoulCave)
```

Format

- `cuzoul.cave.connection` is a 66x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `cuzoul.cave.fragments` is 125x9 data frame:
 - `id`: fragment unique identifier
 - `layer`: stratigraphic layer
 - `sub.layer`: stratigraphic sub-layer
 - `thickness`: thickness of the fragment (in centimetre)
 - `length`: length of the fragment (in centimetre)
 - `surface`: approximated surface of the fragment, product of the thickness by the length (in square centimeter)
 - `x`: x coordinate of the grid square where the bone fragment was found (in centimetre)
 - `y`: y coordinate of the grid square where the bone fragment was found (in centimetre)
 - `z`: approximated z coordinate of the location where the bone fragment was found (in centimetre)

References

- Gardeur M. 2025. Bone refits from the Cuzoul de Gramat Mesolithic layers (archaeological site, France). Zenodo. <https://doi.org/10.5281/zenodo.14975910>

Examples

```
data(CuzoulCave)
head(cuzoul.cave.fragments)
```

CuzoulSouth	<i>Dataset: Refitting relationships between fauna fragments from the Cuzoul de Gramat site</i>
-------------	--

Description

The Cuzoul de Gramat site is a prehistoric site located in France. This dataset describes the refitting 'connection' relationships between bone fragments found during the excavation of Mesolithic layers from the 'South' sector of the site. Connection relationship refers to physical connection between two bone fragments that were part of the same object.

The dataset includes two tables:

- `cuzoul.south.connection`: connection relationships between fragments.
- `cuzoul.south.fragments`: information about the fragments.

Usage

```
data(CuzoulSouth)
```

Format

- `cuzoul.south.connection` is a 135x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `cuzoul.south.fragments` is 233x9 data frame:
 - `id`: fragment unique identifier
 - `layer`: stratigraphic layer
 - `sub.layer`: stratigraphic sub-layer
 - `thickness`: thickness of the fragment (in centimetre)
 - `length`: length of the fragment (in centimetre)
 - `surface`: approximated surface of the fragment, product of the thickness by the length (in square centimeter)
 - `x`: x coordinate of the grid square where the bone fragment was found (in centimetre)
 - `y`: y coordinate of the grid square where the bone fragment was found (in centimetre)
 - `z`: approximated z coordinate of the location where the bone fragment was found (in centimetre)

References

- Gardeur M. 2025. Bone refits from the Cuzoul de Gramat Mesolithic layers (archaeological site, France). Zenodo. <https://doi.org/10.5281/zenodo.14975910>

Examples

```
data(CuzoulSouth)
head(cuzoul.south.fragments)
```

FontJuvenal

Dataset: Refitting relationships between pottery fragments from Font-Juvenal cave

Description

The Font-Juvenal cave is a Neolithic archaeological site in France. This dataset describes the refitting 'connection' relationships between pottery fragments found during excavations. A connection relationship refers to a physical connection between two fragments that were part of the same object.

The dataset is composed of two tables:

- `fontjuvenal.connection`: connection relationships between fragments.
- `fontjuvenal.fragments`: information about the fragments.

Usage

```
data(FontJuvenal)
```

Format

- `fontjuvenal.connection` is a 351x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `fontjuvenal.fragments` is 354x4 data frame:
 - `id`: fragment unique identifier
 - `layer`: simplified stratigraphic layer
 - `layer.detailed`: stratigraphic layer, with detail about where the fragment was found (in the sedimentary layer, or in a hearth contained in the layer)
 - `square`: square where the fragment was found

References

- Caro J. 2024. 'Font-Juvénal_Refiting', Zenodo, doi: <https://doi.org/10.5281/zenodo.14515444>

Examples

```
data(FontJuvenal)
head(fontjuvenal.fragments)
```

frag.cycles	<i>Count the k-cycles in a graph, for cycles =< k</i>
-------------	--

Description

Count the k-cycles in a graph, for cycles =< k

Usage

```
frag.cycles(graph, kmax, max.cycles.only=FALSE)
```

Arguments

graph	An igraph object, must be an undirected graph.
kmax	Maximal length of the cycles to detect.
max.cycles.only	Logical. If TRUE, the fragments are only reported as parts of their longer cycle.

Details

A cycle can be part of larger cycle: if `max.cycles.only` all the cycles are reported but, if this parameter is `True` only the larger cycles are reported. A warning recalls that for cycles $k > 4$ the fragments of a cycle are not necessarily all connected to each other (a fragment, due to its location in the original object, can only be connected to a limited number of adjacent fragments).

Value

A data frame with the number of k-cycles for each k values in [3;k].

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[subgraph_isomorphisms](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
frag.cycles(g, kmax=4, max.cycles.only=FALSE)
frag.cycles(g, kmax=4, max.cycles.only=TRUE)
```

frag.diameters	<i>Diameter distribution for unconnected graphs</i>
----------------	---

Description

Returns the distribution of the diameter values of a fragmentation graph.

Usage

```
frag.diameters(graph, cumulative = FALSE)
```

Arguments

graph	An igraph object.
cumulative	Logical. If TRUE the cumulative relative frequency of the diameters is reported.

Details

frag.diameters wraps the igraph diameter function. For graphs representing the fragmentation of archeological objects, the diameter of each component of the graph (i.e. archeological objects) can be interpreted: as a measure of the intensity of fragmentation (when all the fragments of the initial object are known); as a measure of the scattering of the fragments (when not all the fragments are known);

Value

A numeric vector of the length equal to the maximum diameter value found. The first element is the frequency of the diameter values = 1, the second element is the frequency of diameter values = 2, etc. If cumulative is True, the cumulative density is returned.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[diameter](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
frag.diameters(g)
frag.diameters(g, cumulative=TRUE)
```

frag.edges.weighting *Weighting the edges of a fragmentation graph*

Description

Weighting of the edges of an archeofrag fragmentation graph.

Usage

```
frag.edges.weighting(graph, layer.attr, morphometry, x, y, z, verbose=TRUE)
```

Arguments

graph	An undirected igraph object.
layer.attr	Character. The name of the vertex attribute with the layer of the fragments.
morphometry	Character. Optional, the name of the vertex attribute with the morphometric value of the fragments.
x	Character. Optional, the name of the vertex attribute with the "x" coordinate of the fragments.
y	Character. Optional, the name of the vertex attribute with the "y" coordinate of the fragments.
z	Character. Optional, the name of the vertex attribute with the "z" coordinate of the fragments.
verbose	Logical. Whether to print or not warning messages.

Details

In the framework of the TSAR method, this function weights the edges of a fragmentation graph, before computing the cohesion and admixture values. The weights are computed from the topological properties of the connection network and can be modified using the morphometric properties of the fragments and/or the spatial distance between them. In summary, three different parameters can be used:

1. topology, the basic weighting method;
2. morphometry, the length, length by width, surface, volume, etc.
3. spatial distance, expressed using the metric (or other) system, or an ad hoc relative system.

The function must be applied to a fragmentation graph with two spatial units. Internal connection relationships (within a spatial unit) and external relationships (between the two spatial units) are distinguished, and their respective edge weights are computed in different ways. Three subgraphs are first generated, one for each spatial unit and only one for the external relations.

The weight of an intra-spatial units edge (E) is equal to the sum of the degrees (d) of the vertices (i and j) it connects:

$$W_{intra(E_{ij})} = d_i + d_j$$

For an inter-spatial unit edge, the same calculation is made but with a modifier to account for the balance of information available for each spatial unit:

$$W(E_{interij}) = (d_i + d_j) \times \left(3 - \frac{2}{1 + (tr_i + tr_j)/2}\right) \times \left(1 - \frac{1}{\sqrt{(V_{sub} + E_{sub})}}\right)^2$$

with tr_i and tr_j the local transitivity of the vertices i and j , and $\sqrt{(V_{sub} + E_{sub})}$ the square root of the sum of the vertices count and edge count of the sub-graph.

If the morphometry and/or coordinates (x , y , z) parameters are provided, the previous formula is modified using a factor which is computed as:

$$f(E_{ij}) = 1 - \left(\sqrt{\frac{size_i + size_j}{max(sizes)}} \times \sqrt{\frac{size_i}{size_j} \times \frac{distance_{ij}}{max(distances)}} \right)$$

with $size_i$ the morphometric value of the smaller fragment, $max(sizes)$ the maximum sum of morphometric values observed for the pairs of connected fragments in the dataset under study; $max(prop)$ the maximum proportion between the size values of connected fragments observed in the dataset under study; $distance_{ij}$ the spatial distance between fragments i and j ; $max(distances)$ the maximum euclidean distance observed for the pairs of connected fragments in the dataset under study. Results of the morphometric-spatial factor range in $]0,1]$.

Error messages are displayed if the vertex "layer" attribute has more than two layers, and a warning is displayed if one or more of the values for the x , y , z parameters do not exist in the input graph.

Value

The graph, with an additional "weight" edge attribute and, if the distance has been computed, a "distance" edge attribute.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[transitivity](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
frag.edges.weighting(g, "layer")
# with morphometric and spatial parameters:
library(igraph)
V(g)$morpho <- sample(1:20, 50, replace=TRUE)
V(g)$x <- sample(1:100, 50, replace=TRUE)
V(g)$y <- sample(1:100, 50, replace=TRUE)
V(g)$z <- sample(1:100, 50, replace=TRUE)
frag.edges.weighting(g, "layer", "morpho", "x", "y", "z")
```

frag.get.layers	<i>Extracts the subgraph of each selected archaeological spatial unit.</i>
-----------------	--

Description

Extracts the subgraph of each selected archaeological spatial unit (stratigraphic layer or any other type of spatial unit).

Usage

```
frag.get.layers(graph, layer.attr, sel.layers)
```

Arguments

graph	An undirected igraph object.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of each fragment.
sel.layers	Character. The identifier(s) of the stratigraphic spatial units to retrieve.

Details

This function is only a convenient function to extract the subgraphs of selected archaeological spatial unit (stratigraphic layer or any other type of spatial unit). A graph is created for each layer in the vertex attribute given by the `layer.attr` argument.

Value

A list with a graph for each selected stratigraphic layer.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance = .15)
igraph::V(g)$layers <- c(rep("layer1", 20), rep("layer2", 20), rep("layer3", 10))
frag.get.layers(g, layer.attr="layers", sel.layers=c("layer1", "layer2"))
```

frag.get.layers.pair *Extracts the subgraph corresponding to a pair of archaeological spatial units.*

Description

Extracts the subgraph corresponding to a pair of archaeological spatial units (stratigraphic layers or any other type of spatial unit).

Usage

```
frag.get.layers.pair(graph, layer.attr, sel.layers, size.mini=2,  
                    mixed.components.only=FALSE, verbose=TRUE)
```

Arguments

graph	An igraph object.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of each fragment.
sel.layers	A numeric vector of length 2 with the name of the spatial unit selected for extraction.
size.mini	A minimal number of vertices for the components to include in the resulting graph.
mixed.components.only	Logical. If TRUE, only the components with fragments from the two selected spatial units are returned. If FALSE, all the components of the two spatial units are extracted.
verbose	Logical. Whether to print or not warning messages.

Details

The default setting of the `mixed.components.only` argument is FALSE, for convenience for other measurements.

Value

An undirected graph object.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
igraph::V(g)$layers <- c(rep("layer1", 20), rep("layer2", 20), rep("layer3", 10))

frag.get.layers.pair(g, "layers", sel.layers=c("layer2", "layer3"),
                    size.mini=2, mixed.components.only=FALSE)
frag.get.layers.pair(g, "layers", sel.layers=c("layer2", "layer3"),
                    size.mini=2, mixed.components.only=TRUE)
```

frag.get.parameters *Returns a series of descriptive statistics for a fragmentation graph*

Description

Returns a series of descriptive statistics for a fragmentation graph.

Usage

```
frag.get.parameters(graph, layer.attr, verbose=TRUE)
```

Arguments

graph	An igraph undirected graph.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of each fragment (e.g. layer).
verbose	Logical. Whether to print or not warning messages.

Details

This function is a convenient function to get general information about a fragmentation graph. It is particularly useful for setting the parameters of the `frag.simul.process` function. It returns:

- `n.components`: number of objects (graph connected components)
- `vertices`: number of fragments
- `edges`: number of connection relationships between fragments
- `balance`: estimated fragments balance, i.e. proportion of fragments in the first spatial unit (see details)
- `components balance`: estimated objects balance, i.e. proportion of objects in the first spatial unit (see details)
- `disturbance`: estimated disturbance, i.e. proportion of fragments which moved between spatial units (see details)
- `aggreg.factor`: aggregation factor, diversity of the components' edge count (see details)
- `planar`: whether the graph is planar or not
- `edge.weights.sum`: sum of the edge weights

- `edge.weights.median`: median of the edge weights
- `edge.weights.median.abs.dev`: median absolute deviation of the edge weights

The balance and components balance are calculated from the subgraph including only the fragments connected to fragments from the same spatial unit. This is to approximate these proportions of deposited materials regardless the effects of posterior disturbances. Results are given about the spatial unit whose label comes first alphanumerically (because this function is used internally by the `frag.simul.process` function).

When studying a pair of spatial units, disturbance is defined as the proportion of fragments which moved from their initial spatial unit to the other spatial unit. This definition is used in the `frag.process.simul` function to generate random fragmentation graphs. However, `frag.get.parameters` uses a different definition since its aim is different: considering a fragmentation graph representing the result of unknown post-depositional processes, `frag.get.parameters` returns an estimation of the disturbance which might happened. This estimation is computed from a subgraph including only the components with fragments from the two spatial units: disturbance is get from the number of fragments belonging to the less represented spatial unit in this subgraph over the total number of fragments in the subgraph.

The aggregation factor reflects the diversity of the components' edge count. The factor is calculated by:

$$1 - 1/(1 + sd(components\ edge\ count))$$

The optional RBGL package is required to determine the planarity of the graph. If it is not installed, the `planar` argument is set to `FALSE` by default.

Value

A list of parameters values (`n.components`, `vertices`, `edges`, `balance`, `components.balance`, `disturbance`, `aggreg.factor`, `planar`, `edge.weights.sum`, `edge.weights.median`, `edge.weights.median.abs.dev`).

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[frag.get.layers.pair](#), [frag.simul.process](#), [sd](#), [median](#), [mad](#), [boyerMyrvoldPlanarityTest](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=0.1)
frag.get.parameters(g, "layer")
```

frag.graph.plot *Plot a fragmentation graph*

Description

A function to plot the graph made by the archeofrag package.

Usage

```
frag.graph.plot(graph, layer.attr, ...)
```

Arguments

graph	An igraph undirected object with a "frag_type" attribute.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of each fragment.
...	Optional arguments sent to plot.igraph.

Details

This function is a wrapper for the plot.igraph method for igraph objects. The layout is computed using the fruchterman-reingold algorithm, with some changes as a function of the value of the "frag_type" graph attribute. For graphs including similarity relations, igraph' component_wise layout modifier is applied. For graphs with connection and similarity relationships, the edges for connection relations are coloured in green. For graphs with connection relationships only and two spatial units, the vertices are placed on the top (spatial unit 1) and bottom (spatial unit 2) part of the plot.

Value

Returns NULL and plot the graph.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[plot.igraph](#), [component_wise](#), [layout_with_fr](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
frag.graph.plot(g, "layer")
```

frag.graph.reduce *Reduce the size of a fragmentation graph*

Description

Remove fragments from a fragmentation graph

Usage

```
frag.graph.reduce(graph=NULL, n.frag.to.remove=NULL, conserve.objects.nr=FALSE)
```

Arguments

graph	An igraph object.
n.frag.to.remove	Integer. Number of fragments (i.e. vertices) to remove.
conserve.objects.nr	Logical. If TRUE, preserve the number of objects (i.e. connected components) in the graph.

Details

This function reduces the number of fragments in a fragmentation graph, while controlling that no singletons are created, and (optionally) that the output graph and the input graph have the same number of connected components. Note that if 'conserve.objects.nr' is TRUE then the reduction process can stop before reaching the number of fragments to remove (when the graph only contains pairs of fragments).

Value

A fragmentation graph (igraph object) with less fragments (vertices).

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

Examples

```
g <- frag.simul.process(n.components=15, vertices=50, disturbance = .15)
igraph::gorder(g)
igraph::components(g)$no
# reduce the number of fragments and conserve the number of connected components:
g1 <- frag.graph.reduce(g, n.frag.to.remove = 40, conserve.objects.nr = TRUE)
igraph::gorder(g1)
igraph::components(g1)$no
# reduce the number of fragments and do not conserve the number of connected components:
g2 <- frag.graph.reduce(g, n.frag.to.remove = 40, conserve.objects.nr = FALSE)
igraph::gorder(g2)
igraph::components(g2)$no
```

frag.layers.admixture *Admixture of two spatial units*

Description

Evaluate how reliable the distinction is between two archaeological spatial units (layers or any other type of spatial units).

Usage

```
frag.layers.admixture(graph, layer.attr, morphometry, x, y, z, verbose=TRUE)
```

Arguments

graph	An undirected igraph object.
layer.attr	Character. The name of the vertex attribute giving the spatial unit of each fragment (e.g. layer).
morphometry	Character. Optional, to pass to the 'frag.edges.weighting' function: name of the vertex attribute with the morphometric value of the fragments.
x	Character. Optional, to pass to the 'frag.edges.weighting' function: name of the vertex attribute with the "x" coordinate of the fragments.
y	Character. Optional, to pass to the 'frag.edges.weighting' function: name of the vertex attribute with the "y" coordinate of the fragments.
z	Character. Optional, to pass to the 'frag.edges.weighting' function: name of the vertex attribute with the "z" coordinate of the fragments.
verbose	Logical. Whether to print or not warning messages.

Details

This function returns a value reflecting the robustness of the distinction between two spatial units (layers or any other relevant archaeological spatial unit). The admixture value is computed as:

$$1 - \text{cohesion}(\text{layer } 1) - \text{cohesion}(\text{layer } 2)$$

The admixture of two spatial units is equal to the cohesion of a virtual third unit, which is defined by the fragments and the connection relationships intersecting the two spatial units. Results range in [0;1] with 0 for two completely independent spatial units and values towards 1 as the robustness of the boundary between the two spatial units is lower. As it appears, this function calls the frag.layers.cohesion function.

The basic use of this function is intended for a graph with two layers, whose edges have been previously weighted using the frag.edges.weighting function (an error message is displayed if the vertex attribute "layer" contains less than two spatial units, and if the graph does not have an edge attribute "weight").

However, the function can also be used for a graph with more than two spatial units. In this case, a subgraph is generated for each pair of layers (using the frag.get.layers.pair function), the frag.edges.weighting function is applied to weight their edges (a warning message is displayed), and the admixture is computed.

An error message is displayed if the vertex "layer" attribute has less than two values.

Value

A numeric vector with the admixture of each pair of spatial units.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[frag.edges.weighting](#), [frag.layers.cohesion](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance = .15)
g <- frag.edges.weighting(g, layer.attr="layer")
frag.layers.admixture(g, "layer")
```

frag.layers.cohesion *Cohesion measure of layers*

Description

Returns the cohesion value of two archaeological spatial units (layers or any other type of spatial units).

Usage

```
frag.layers.cohesion(graph, layer.attr, morphometry, x, y, z, verbose=TRUE)
```

Arguments

graph	An undirected igraph object.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of the fragments.
morphometry	Character. Optional, to pass to the ‘frag.edges.weighting’ function: name of the vertex attribute with the morphometric value of the fragments.
x	Character. Optional, to pass to the ‘frag.edges.weighting’ function: name of the vertex attribute with the "x" coordinate of the fragments.
y	Character. Optional, to pass to the ‘frag.edges.weighting’ function: name of the vertex attribute with the "y" coordinate of the fragments.
z	Character. Optional, to pass to the ‘frag.edges.weighting’ function: name of the vertex attribute with the "z" coordinate of the fragments.
verbose	Logical. Whether to print or not warning messages.

Details

Note that this function must be used after weighting the edges with `frag.edges.weighting`.

The cohesion value of a spatial unit is computed as:

$$\frac{V_{unit_i} + W_{unit_i}}{\sum_{j=1}^2 V_{unit_j} + W_{unit_j}}$$

with V the number of vertices in the unit and W the sum of the edge weights within the unit.

The measure takes into account the balance between the information about each spatial unit. Results range in $]0;1[$, with 0 for two spatial units with only inter-spatial units connection relationships, and 1 if there are not inter-spatial units relationships and a significant imbalance of information on the two spatial units.

The basic use of this function is intended for a graph with two spatial units, whose edges have been previously weighted using the `frag.edges.weighting` function (an error message is displayed if the vertex attribute "layer" contains less than two spatial units, and if the graph does not have an edge attribute "weight").

However, this function can also be used for a graph with more than two spatial units. In this case, a subgraph is generated for each pair of spatial units (using the `frag.get.layers.pair` function), the `frag.edges.weighting` function is applied to weight their edges (a warning message is displayed), and the cohesion is computed.

Value

If the graph has only two spatial units, the function returns a numeric vector with a cohesion value ($]0;1[$) for each spatial units (sorted in alphanumerical order). If the graph has more than two spatial units, the function returns a matrix with cohesion values for each pair of spatial units.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[frag.edges.weighting](#), [frag.get.layers.pair](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.1)
frag.layers.cohesion(g, layer.attr="layer")
```

Frag.object-class	Class "Frag.object"
-------------------	---------------------

Description

A class for archaeological "fragmentation" datasets. This class construction aims to ensure that the data have been properly built before performing the next steps of the analysis. A convenient constructor function, [make_frag_object](#), is provided.

Objects from the Class

Objects can be created by calls of the form:

```
make_frag_object(cr, sr, fragments).
```

Slots

df.cr: Object of class "matrix" ("data.frame" are allowed and automatically converted)

df.sr: Object of class "matrix" ("data.frame" are allowed and automatically converted)

fragments.df: Object of class "data.frame"

frag_type: Object of class "character"

Methods

make_cr_graph signature(object = "Frag.object"): Makes an undirected graph representing the "connection" relationships between archaeological fragments.

make_sr_graph signature(object = "Frag.object"): Makes an undirected graph representing the "similarity" relationships between archaeological fragments.

make_crsr_graph signature(object = "Frag.object"): Makes an undirected graph combining the "connection" and "similarity" relations between archaeological fragments.

show signature(object = "Frag.object"): show method for Frag.object

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[make_frag_object](#), [make_cr_graph](#), [make_sr_graph](#), [make_crsr_graph](#)

Examples

```
showClass("Frag.object")
```

frag.observer.failure *Simulate the failure of an observer to determine the relationships between fragments.*

Description

Simulate the failure of an observer to determine the relationships between fragments.

Usage

```
frag.observer.failure(graph, likelihood, remove.vertices=FALSE)
```

Arguments

graph	An undirected igraph object.
likelihood	Numerical vector of values in [0,1] giving the likelihood of not observing a relationship between two fragments.
remove.vertices	Logical. If TRUE, unconnected vertices are removed.

Details

In determining connection relationships between archaeological fragments, archaeologists often consider the likelihood that they fail in identifying some of these relationships. Given an initial fragmentation graph, this function aims to simulate the effects of such different likelihood values.

For each value in the likelihood parameter, a new graph is generated by randomly removing the given proportion of edges from the input graph. When generating series of comparable graphs with different likelihoods, the function internally resets the seed for random number generation, to ensure that edges are removed in the same order.

An error message is displayed if at least one of the likelihood values is < 0 or > 1 .

Value

A list of fragmentation graphs (igraph objects).

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[set.seed](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
frag.observer.failure(graph=g, likelihood=c(0.05, 0.1), remove.vertices=FALSE)
```

frag.path.lengths *Path length distribution for unconnected graphs*

Description

Path length distribution for unconnected graphs

Usage

```
frag.path.lengths(graph, cumulative=FALSE)
```

Arguments

graph	An igraph object.
cumulative	Logical. If TRUE, the cumulative relative frequency of the path lengths is returned.

Details

This function is a wrapper of `igraph distance_table` returning the frequency of path lengths in undirected and unconnected graphs. In the context of archaeological fragmentation analysis, path lengths are interpreted to characterise the properties of fragmentation within a layer.

Value

A numeric vector having the same length as the maximum path length. The first element of the vector is the frequency of the paths of length 1, the second element is the frequency of the paths of length 2, etc.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[distance_table](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance = .15)
frag.path.lengths(g)
frag.path.lengths(g, cumulative=TRUE)
```

frag.relations.by.layers

Summary of the connection relationships between fragments within and between spatial units.

Description

Return a matrix with the number of relationships within and between spatial units (e.g., layers).

Usage

```
frag.relations.by.layers(graph, layer.attr)
```

Arguments

graph	An igraph object.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of each fragment.

Details

This function is a useful method to summarise the distribution of connection relationships within and between spatial units (e.g., layers).

Value

A square matrix with the number of connection relationships within and between spatial units.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
frag.relations.by.layers(g, "layer")
```

frag.simul.compare	<i>From an observed fragmentation graph, simulates two series of graphs corresponding to two deposition hypotheses.</i>
--------------------	---

Description

Given an observed fragmentation graph, simulates two series of graphs corresponding to two deposition hypotheses, compares their properties and returns a summary table.

Usage

```
frag.simul.compare(graph, layer.attr, iter, summarise=TRUE, verbose=TRUE, ...)
```

Arguments

graph	An undirected igraph object. The 'observed' graph to compare to simulated graphs.
layer.attr	Character. The name of the vertices attribute giving the layer of the fragments.
iter	Numerical. The number of simulated graphs to generate for each hypothesis (minimal value: 30).
summarise	Logical. Whether to report a comparative summary of the results.
verbose	Logical. Whether to print or not warning messages.
...	Further arguments passed to the 'frag.simul.process' function.

Details

This function is a convenient wrapper integrating several functions of the archeofrag package to compare an observed fragmentation graph to similar simulated graphs. The `frag.simul.process` is used to generate two series of graphs from the properties of the observed graph: the first series is generated under the formation hypothesis H1 (one initial spatial unit) and the second series is generated under the hypothesis H2 (two initial spatial units). The edge count, edge weights sum, balance, components balance, disturbance, admixture, and cohesion values of the generated graphs are measured.

By default, the results are post-processed with the `frag.simul.summarise` function and a summary data frame is printed and included in the list of results which is silently returned. If the `summarise` parameter is set to `FALSE`, then the function returns a list of two data frames containing the numeric values measured for H1 and H2.

Value

A named list with three items: "h1.data", a data frame with the numerical values measured on the graphs generated for H1; "h2.data", a data frame with the numerical values measured on the graphs generated for H2; "summary", a data frame summarising the comparison between the results for the two hypotheses and the values measured on the empirical graph.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[frag.simul.process](#), [frag.simul.summarise](#)

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
## Not run: frag.simul.compare(g, layer.attr="layer", iter=30)
```

frag.simul.process	<i>Simulate the fragmentation of archaeological objects scattered in two spatial units</i>
--------------------	--

Description

Simulate the fragmentation of archaeological objects scattered in two spatial units.

Usage

```
frag.simul.process(initial.layers=2, n.components=NULL, vertices=Inf,
                  edges=Inf, balance=.5, components.balance=.5,
                  disturbance=0, aggreg.factor=0, planar=FALSE,
                  asymmetric.transport.from=NULL,
                  from.observed.graph=NULL, observed.layer.attr=NULL)
```

Arguments

initial.layers	Integer (1 or 2). Number of hypothetical spatial units (e.g. layers) to use as initial condition.
n.components	Integer. Initial number of objects (connected components).
vertices	Integer. Final number of fragments (vertices).
edges	Integer. Final number of connection relationships between fragments (edges).
balance	Numeric]0;1[. Proportion of fragments to generate in the first spatial unit before applying disturbance.
components.balance	Numeric]0;1[. Proportion of objects (connected components) in the first spatial unit before applying fragmentation (only used when initial.layers=2).
disturbance	Numeric [0;1]. Final proportion of fragments moved from a spatial unit to the other.
aggreg.factor	Numeric [0;1]. Higher values increase the likelihood that the biggest components are selected when adding fragments or connections.

<code>planar</code>	Logical. If TRUE, generates a planar graph (if FALSE, the graph can be planar or not).
<code>asymmetric.transport.from</code>	Numeric or character value in "1" or "2" (referring to the first and second spatial unit). If not NULL, the disturbance process will be applied only to the fragments from this spatial unit.
<code>from.observed.graph</code>	igraph object. If not NULL, the parameters observed in this fragmentation graph are used instead of the previous parameters. See details.
<code>observed.layer.attr</code>	character. Required if the <code>from.observed.graph</code> option is used. Name of the spatial unit attribute in the observed graph.

Details

This function simulates the fragmentation of archeological objects within and between two adjacent stratigraphic layers. Fragments are represented by vertices and the "connection" relationships ("re-fittings") between them are represented by edges. All fragments have at least one relation ("single" fragments are not generated).

Some parameters are optional or depend on other parameters (messages are displayed accordingly). Namely, setting `initial.layers=1` enables to constraint the graph with the number of vertices only, the number of edges only, or both. With `initial.layers=2`, the `components.balance` can be used, and the `edges` parameter is not supported (only the `vertices` parameter can be used).

The `components.balance` parameter determines the proportion of components (i.e. objects) in the first spatial unit before the application of the fragmentation process; the `balance` parameter determines the proportion of fragments in the first spatial unit before the application of the disturbance process. The `disturbance` parameter determines the proportion of fragments to move from one spatial unit to another. Consequently, it generates inter-spatial units relationships. Highest admixture are generated with `disturbance=0.5`. If `asymmetric.transport.from` is set to 1 or 2, the disturbance process is only applied to the fragments from spatial unit 1 or spatial unit 2, respectively.

If a graph is given to the `from.observed.graph` parameter, the properties of this graph are internally retrieved with the `frag.get.parameters` function (including: the number of components, number of vertices, `balance`, the `components.balance`, the `disturbance`, the `aggregation.factor`, and whether the graph is planar or not; note that the number of edges is not included as a parameter). If some other parameters of the `frag.simul.process` function are set, the values retrieved from the observed graph are used in replacement. The `frag.edges.weighting` is internally applied to weight the graph edges.

Setting the `planar` argument to TRUE constraints the graph to be planar (if this parameter is FALSE, the graph can be planar or not). Note that using the `planar` argument requires to install the optional RBGL package and that the simulator is faster with `initial.layers=2` and `planar=FALSE`.

Value

An igraph object with a `"frag_type"` graph attribute (with the value "cr", for "connection relationship") and three vertices attributes: `"name"` (vertices identifiers), `"layer"` (with the values "1" and "2"), and `"object.id"` (component identifiers).

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[frag.get.parameters](#), [frag.edges.weighting](#), [boyerMyrvoldPlanarityTest](#)

Examples

```
frag.simul.process(n.components=20, vertices=50, disturbance=.15)

g <- frag.simul.process(initial.layers=1,
                       n.components=20,
                       vertices=50,
                       edges=40,
                       balance=.5,
                       components.balance=.5,
                       disturbance=.1,
                       planar=FALSE)
plot(g, vertex.color=factor(igraph::V(g)$layer),
     vertex.size=4, vertex.label=NA)
```

frag.simul.summarise *Summarise the comparison between an observed fragmentation graph and simulated graphs for two deposition hypotheses.*

Description

Compare the parameters measured on simulated graphs generated for two deposition hypotheses with Wilcoxon tests, and the corresponding values measured on an empirical graph, and returns a summary data frame.

Usage

```
frag.simul.summarise(graph, layer.attr, res.h1, res.h2, cohesion1.attr="cohesion1",
                    cohesion2.attr="cohesion2", admixture.attr="admixture",
                    verbose=TRUE)
```

Arguments

graph	An undirected igraph object. The graph to compare with simulated graphs.
layer.attr	Character. The name of the vertices attribute giving the spatial unit of the fragments (layer or other).
res.h1	data frame. A data frame with the parameters observed on the simulated graphs for H1.
res.h2	data frame. A data frame with the parameters observed on the simulated graphs for H2.

cohesion1.attr	character. The name of the column in the data frames res.h1 and res.h2 with the cohesion values of the spatial unit 1.
cohesion2.attr	character. The name of the column in the data frames res.h1 and res.h2 with the cohesion values of the spatial unit 2.
admixture.attr	character. The name of the column in the data frames res.h1 and res.h2 with the admixture values.
verbose	Logical. Whether to print or not warning messages.

Details

This function compares and summarises the numerical values measured on the series of graphs generated for the two deposition hypotheses (H1: one initial spatial unit, H2: two initial spatial units). It is intended to post-process the results of the `frag.simul.compare` function, but it can also be applied to other inputs.

The data frames with the results for H1 and H2 must have exactly the same column names, corresponding to parameters measured on the simulated graphs. When using the result generated with the `frag.simul.compare` function, the parameters considered include: the edge count, the sum of the edge weights, the balance, the components balance, the disturbance, the admixture and the cohesion of the two spatial units). When using this function alone, the names of the columns of the data frames can be set with the `cohesion1.attr`, `cohesion2.attr`, and `admixture.attr` parameters (default values: "cohesion1", "cohesion2", "admixture" respectively).

For each parameter, a two-sample Wilcoxon test is run to compare the series of values generated for H1 and H2. In addition, the value measured on the observed graph is compared with the range of values generated for the two hypotheses. The results of these comparisons are reported as a data frame with four columns: for each parameter studied, the data frame contains 1) whether the series of H1 values are statistically different to the H2 series (Boolean), 2) the p-value of the Wilcoxon test (numerical), 3) whether the observed value is "within", "higher", or "lower" to the interquartile range of values for H1, 4) whether the observed value is "within", "higher", or "lower" to the interquartile range of values for H2.

Value

A data frame summarising the comparison between the simulated results for the two hypotheses, and the values measured on the empirical graph with the simulated results.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[frag.simul.compare](#), [wilcox.test](#),

Examples

```
g <- frag.simul.process(n.components=20, vertices=50, disturbance=.15)
g <- frag.edges.weighting(g, layer.attr="layer")
## Not run: res <- frag.simul.compare(g, layer.attr="layer", iter=30, summarise=FALSE)
```

```
frag.simul.summarise(g, layer.attr="layer", res.h1=res[[1]], res.h2=res[[2]])  
## End(Not run)
```

GrandeRivoire	<i>Dataset: Refitting relationships between lithic fragments from the Grande Rivoire site</i>
---------------	---

Description

Grande Rivoire is a Mesolithic rock shelter located in France. This dataset describes the refitting 'connection' relationships between lithic fragments found during excavations. Connection relationship refers to physical connection between two fragments that were part of the same object.

The dataset is composed of two tables:

- `grande.rivoire.connection`: connection relationships between fragments.
- `grande.rivoire.fragments`: information about the fragments.

Usage

```
data(GrandeRivoire)
```

Format

- `grande.rivoire.connection` is a 71x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `grande.rivoire.fragments` is a 91x3 data frame:
 - `id`: fragment unique identifier
 - `layer`: archaeological layer where the fragment was found
 - `layer.su`: combination of the archaeological layer and sedimentary unit where the fragment was found

References

- Angelin A. 2025. 'Refitting data from La Grande Rivoire prehistoric site', Zenodo, doi: <https://doi.org/10.5281/zenodo.14609875>

Examples

```
data(GrandeRivoire)  
head(grande.rivoire.fragments)
```

LiangAbu

Dataset: Archeological relationships between pottery fragments in Liang Abu

Description

Liang Abu is an archaeological site in East Kalimantan, Indonesia. This dataset describes the relationships between pottery fragments found during excavations (2009-2012). Two types of relationships are defined.

- A connection relationship refers to a physical connection between two fragments that were part of the same object.
- A similarity relationship between fragments is defined if there is an acceptable likelihood that those fragments were part of the same object.

The dataset is composed of three tables:

- `liangabu.connection`: "connection" relationships between fragments.
- `liangabu.similarity`: "similarity" relationships between fragments.
- `liangabu.fragments`: information about the fragments.

Usage

```
data(LiangAbu)
```

Format

- `liangabu.connection` is a 56x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id', respectively.
- `liangabu.similarity` is a 147x2 matrix. Column 'id' gives a fragment unique identifier, column 'su.id' gives a unique identifier for the group of similar fragments it belongs to (similarity unit).
- `liangabu.fragments` is 177x11 data frame:
 - `id`: fragment unique identifier
 - `layer`: stratigraphic layer
 - `zmin`: minimal depth in centimetres where the fragment was found
 - `zmax`: maximal depth in centimetres where the fragment was found
 - `square`: grid square where the fragment was found
 - `square.x`: x coordinate in centimetres of the middle of the grid square
 - `square.y`: y coordinate in centimetres of the middle of the grid square
 - `thickness`: thickness of the fragments in millimetres
 - `length`: length of the fragments in millimetres
 - `thickness.by.length`: product of the thickness by the length
 - `sherd.type`: type of pottery sherd

References

- Plutniak, Sebastien, "The Strength of Parthood Ties. Modelling Spatial Units and Fragmented Objects with the TSAR Method - Topological Study of Archaeological Refitting", Journal of Archaeological Science, vol. 136, 105501, doi: 10.1016/j.jas.2021.105501.
- Plutniak, Sebastien, "Refitting pottery fragments from the Liang Abu rockshelter, Borneo", Zenodo, doi: 10.5281/zenodo.4719578.
- Plutniak, Sebastien, Astolfo Araujo, Simon Puaud, Jean-Georges Ferrie, Adhi Agus Oktaviana, Bambang Sugiyanto, Jean-Michel Chazine et Francois-Xavier Ricaut. 2015. "Borneo as a half empty pot: Pottery assemblage from Liang Abu, East Kalimantan, Quaternary International, doi: 10.1016/j.quaint.2015.11.080.

Examples

```
data(LiangAbu)
head(liangabu.fragments)
```

make_crsr_graph	<i>Makes a "connection" relationships graph including the "similarity" relationships.</i>
-----------------	---

Description

Takes a `frag.object` in argument and returns an undirected graph representing the relationships between archaeological fragments. "Connection" and "similarity" relationships are combined. A "connection" relationship refers to a physical connection between two fragments that were part of the same object. A "similarity" relationship between fragments is defined if there is an acceptable likelihood that those fragments were part of the same object.

Usage

```
make_crsr_graph(object)
```

Arguments

`object` A `frag.object`.

Details

A complementary function to the `make_cr_graph` function. This function handles both the "connection" and "similarity" relationships. This can be useful, given that "similarity" relations are more frequently documented in archaeological datasets than the "connection" relationships.

The function returns an undirected graph of "igraph" class, using the "fragments" data frame of the `frag.object` to set the vertices attributes.

Both "connection" and "similarity" relationships are included in the resulting graph. The edge attribute "type_relation" is set with a character "cr" value for "connection" relationships and with "sr" for "similarity" relationships. Edge weights are not set by this function, and it is recommended to use the `frag.edges.weighting` function. A "frag_type" graph attribute is set with a "connection and similarity" value.

Value

An undirected "igraph" class graph.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[make_frag_object](#), [make_cr_graph](#), [make_sr_graph](#)

Examples

```
cr.df <- matrix(c(1,2, 1,3, 2,3, 4,5, 4,6, 7,8), ncol=2, byrow=TRUE)
sr.df <- matrix( c(1,1, 9,1, 10,1, 11,2, 12,2, 13,2), ncol=2, byrow=TRUE)
fragments.df <- data.frame(1:13, letters[1:13])
crsr_g <- make_frag_object(cr=cr.df, sr=sr.df, fragments=fragments.df)
make_crsr_graph(crsr_g)
```

make_cr_graph

Make a "connection" relationships graph.

Description

Takes a [frag.object](#) and returns an undirected graph representing the "connection" relationships between archaeological fragments. A "connection" relationship refers to a physical connection between two fragments that were part of the same object.

Usage

```
make_cr_graph(object)
```

Arguments

object A [frag.object](#) object.

Details

Returns an undirected graph of "igraph" class. The "fragments" data frame of the frag.object is used to set the vertices attributes.

Value

An undirected igraph class graph. The "frag_type" graph attribute is set with the "connection" character value.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[make_frag_object](#)

Examples

```
cr.df <- matrix(c(1,2, 1,3, 2,3, 4,5, 4,6, 7,8), ncol=2, byrow=TRUE)
sr.df <- matrix( c(1,1, 9,1, 10,1, 11,2, 12,2, 13,2), ncol=2, byrow=TRUE)
fragments.df <- data.frame(1:13, letters[1:13])

cr_g <- make_frag_object(cr=cr.df, fragments=fragments.df)
crsr_g <- make_frag_object(cr=cr.df, sr=sr.df, fragments=fragments.df)

make_cr_graph(cr_g)
make_cr_graph(crsr_g)
```

make_frag_object	Makes a "frag.object" object.
------------------	-------------------------------

Description

Makes a "[frag.object](#)" object.

Usage

```
make_frag_object(cr, sr, fragments)
```

Arguments

cr	A matrix or a data frame with two columns giving the vertex id of each pair of connected fragments.
sr	Optional. A matrix or a data frame with two columns: the first gives the fragment id, the second gives the "similarity group" id. Optional if mode is "cr".
fragments	A matrix or a data frame with information about each fragment. The first column must contain the fragments' id.

Details

This function checks the dataset and returns a "frag.object" which can be turned into a fragmentation graph using the [make_cr_graph](#), [make_sr_graph](#), or [make_crsr_graph](#) functions.

Value

An object of "[frag.object](#)" class.

Author(s)

Sebastien Plutniak <sebastien.plutniak@posteo.net>

Examples

```
cr.df <- matrix(c(1,2, 1,3, 2,3, 4,5, 4,6, 7,8), ncol=2, byrow=TRUE)
sr.df <- matrix( c(1,1, 9,1, 10,1, 11,2, 12,2, 13,2), ncol=2, byrow=TRUE)
fragments.df <- data.frame(1:13, letters[1:13])

make_frag_object(cr=cr.df, fragments=fragments.df)
make_frag_object(cr=cr.df, sr=sr.df, fragments=fragments.df)
```

make_sr_graph

Make a "similarity" relationships graph.

Description

Takes a [frag.object](#) and returns an undirected graph representing the "similarity" relationships between archaeological fragments. A "similarity" relationship between fragments is defined if there is an acceptable likelihood that those fragments were part of the same object.

Usage

```
make_sr_graph(object)
```

Arguments

object A [frag.object](#) object.

Details

Returns an undirected graph of "igraph" class. The "fragments" data frame of the frag.object is used to set the vertices attributes.

Value

An undirected igraph class graph. The "frag_type" graph attribute is set with the "similarity" character value.

Author(s)

Sebastien Plutniak <sebastien.plutniak at posteo.net>

See Also

[make_frag_object](#)

Examples

```
sr.df <- matrix( c(1,1, 9,1, 10,1, 11,2, 12,2, 13,2), ncol=2, byrow=TRUE)
fragments.df <- data.frame(1:13, letters[1:13])
crsr_g <- make_frag_object(sr=sr.df, fragments=fragments.df)
make_sr_graph(crsr_g)
```

TaiCave

Dataset: Refitting relationships between pottery fragments from the Tai site, Cave sector

Description

The Tai is a Neolithic archaeological site in France. Three sectors were distinguished: the North entrance, the South entrance and the Cave. This dataset regards the refitting 'connection' relationships between pottery fragments found during excavations led by Claire Manen in the Cave sector. A connection relationship refers to a physical connection between two fragments that were part of the same object.

The dataset includes two tables:

- `tai.cave.connection`: connection relationships between fragments.
- `tai.cave.fragments`: information about the fragments.

Usage

```
data(TaiCave)
```

Format

- `tai.cave.connection` is a 91x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `tai.cave.fragments` is 112x7 data frame:
 - `id`: fragment unique identifier
 - `layer`: stratigraphic layer
 - `sherd_size`: size class of the pottery sherd (nb: the numeric value is not metrical)
 - `sherd_thickness`: thickness of the pottery sherd in millimeter
 - `x`: x coordinates of the location where the pottery sherd was found (in centimetres)
 - `y`: y coordinates of the location where the pottery sherd was found (in centimetres)
 - `z`: z coordinates of the location where the pottery sherd was found (in centimetres)

References

- Caro J., Plutniak S. 2022. 'Refitting and Matching Neolithic Pottery Fragments from the Tai site, France'. Zenodo, doi: <https://doi.org/10.5281/zenodo.7408706>

Examples

```
data(TaiCave)
head(tai.cave.fragments)
```

TaiSouth	<i>Dataset: Refitting relationships between pottery fragments from the Tai site, South entrance sector</i>
----------	--

Description

The Tai is a Neolithic archaeological site in France. Three sectors were distinguished: the North entrance, the South entrance and the Cave. This dataset regards the refitting 'connection' relationships between pottery fragments found during excavations led by Claire Manen in the South entrance sector. A connection relationship refers to a physical connection between two fragments that were part of the same object.

The dataset includes two tables:

- `tai.south.connection`: connection relationships between fragments.
- `tai.south.fragments`: information about the fragments.

Usage

```
data(TaiSouth)
```

Format

- `tai.south.connection` is a 79x2 matrix. Each line describes a connection relationship between two fragments. Their unique identifiers are given in column 'id1' and in column 'id2', respectively.
- `tai.south.fragments` is 82x7 data frame:
 - `id`: fragment unique identifier
 - `layer`: stratigraphic layer
 - `sherd_size`: size class of the pottery sherd (nb: the numeric value is not metrical)
 - `sherd_thickness`: thickness of the pottery sherd in millimeter
 - `x`: x coordinates of the location where the pottery sherd was found (in centimetres)
 - `y`: y coordinates of the location where the pottery sherd was found (in centimetres)
 - `z`: z coordinates of the location where the pottery sherd was found (in centimetres)

References

- Caro J., Plutniak S. 2022. 'Refitting and Matching Neolithic Pottery Fragments from the Tai site, France'. Zenodo, doi: <https://doi.org/10.5281/zenodo.7408706>

Examples

```
data(TaiSouth)
head(tai.south.fragments)
```

Index

- * **build data**
 - frag.edges.weighting, 10
 - Frag.object-class, 21
 - make_cr_graph, 33
 - make_crsr_graph, 32
 - make_frag_object, 34
 - make_sr_graph, 35
- * **datasets**
 - BoutDesVergnes, 3
 - Chauzeys, 4
 - CuzoulCave, 5
 - CuzoulSouth, 6
 - FontJuvenal, 7
 - GrandeRivoire, 30
 - LiangAbu, 31
 - TaiCave, 36
 - TaiSouth, 37
- * **extraction**
 - frag.get.layers, 12
 - frag.get.layers.pair, 13
- * **measurement**
 - frag.cycles, 8
 - frag.diameters, 9
 - frag.layers.admixture, 18
 - frag.layers.cohesion, 19
 - frag.path.lengths, 23
- * **package**
 - archeofrag-package, 2
- * **simulation**
 - frag.graph.reduce, 17
 - frag.observer.failure, 22
 - frag.simul.compare, 25
 - frag.simul.process, 26
 - frag.simul.summarise, 28
- * **summary**
 - frag.get.parameters, 14
 - frag.graph.plot, 16
 - frag.relations.by.layers, 24
- archeofrag (archeofrag-package), 2
- archeofrag-package, 2
- bdv.connection (BoutDesVergnes), 3
- bdv.fragments (BoutDesVergnes), 3
- BoutDesVergnes, 3
- boyerMyrvoldPlanarityTest, 15, 28
- Chauzeys, 4
- chauzeys.connection (Chauzeys), 4
- chauzeys.fragments (Chauzeys), 4
- component_wise, 16
- cuzoul.cave.connection (CuzoulCave), 5
- cuzoul.cave.fragments (CuzoulCave), 5
- cuzoul.south.connection (CuzoulSouth), 6
- cuzoul.south.fragments (CuzoulSouth), 6
- CuzoulCave, 5
- CuzoulSouth, 6
- diameter, 9
- distance_table, 23
- FontJuvenal, 7
- fontjuvenal.connection (FontJuvenal), 7
- fontjuvenal.fragments (FontJuvenal), 7
- frag.cycles, 8
- frag.diameters, 9
- frag.edges.weighting, 10, 19, 20, 28
- frag.get.layers, 12
- frag.get.layers.pair, 13, 15, 20
- frag.get.parameters, 14, 28
- frag.graph.plot, 16
- frag.graph.reduce, 17
- frag.layers.admixture, 18
- frag.layers.cohesion, 19, 19
- frag.object, 32–35
- Frag.object-class, 21
- frag.observer.failure, 22
- frag.path.lengths, 23
- frag.relations.by.layers, 24
- frag.simul.compare, 25, 29

- frag.simul.process, [15](#), [26](#), [26](#)
- frag.simul.summarise, [26](#), [28](#)

- grande.rivoire.connection
 (GrandeRivoire), [30](#)
- grande.rivoire.fragments
 (GrandeRivoire), [30](#)
- GrandeRivoire, [30](#)

- igraph, [3](#)

- layout_with_fr, [16](#)
- LiangAbu, [31](#)
- liangabu.connection (LiangAbu), [31](#)
- liangabu.fragments (LiangAbu), [31](#)
- liangabu.similarity (LiangAbu), [31](#)

- mad, [15](#)
- make_cr_graph, [21](#), [32](#), [33](#), [33](#), [34](#)
- make_cr_graph, Frag.object-method
 (Frag.object-class), [21](#)
- make_cr_graph-methods (make_cr_graph),
 [33](#)
- make_csr_graph, [21](#), [32](#), [34](#)
- make_csr_graph, Frag.object-method
 (Frag.object-class), [21](#)
- make_frag_object, [21](#), [33](#), [34](#), [34](#), [35](#)
- make_sr_graph, [21](#), [33](#), [34](#), [35](#)
- make_sr_graph, Frag.object-method
 (Frag.object-class), [21](#)
- make_sr_graph-methods (make_sr_graph),
 [35](#)
- median, [15](#)

- plot.igraph, [16](#)

- RBGL, [3](#)

- sd, [15](#)
- set.seed, [22](#)
- show, Frag.object-method
 (Frag.object-class), [21](#)
- subgraph_isomorphisms, [8](#)

- tai.cave.connection (TaiCave), [36](#)
- tai.cave.fragments (TaiCave), [36](#)
- tai.south.connection (TaiSouth), [37](#)
- tai.south.fragments (TaiSouth), [37](#)
- TaiCave, [36](#)
- TaiSouth, [37](#)

- transitivity, [11](#)

- wilcox.test, [29](#)