

Package: netCoin (via r-universe)

October 26, 2026

Type Package

Version 2.1.47

Date 2026-10-23

Title Interactive Analytic Networks

Description Combines the data analysis capabilities of R with the interactive visualization libraries of JavaScript to create networks of coincidences, co-occurrences, correlations and regressions that can be explored directly in a browser, embedded in HTML files or integrated into Shiny applications. It also generates networked HTML galleries and other plots of coincidences.

License GPL-2 | GPL-3

Depends R (>= 3.5.0)

Imports rD3plot (>= 1.1.57), igraph (>= 1.0.1), Matrix (>= 1.2-4), haven (>= 1.1.0), MASS (>= 7.3), GPArotation(>= 2022.4), marginaleffects, methods, nnet

Suggests knitr, rmarkdown, lavaan, shiny, htmltools, openxlsx, data.table, readr, tidyLPA, clue, NMF, Rtsne, cluster, dendextend, fpc, parallelDist, uwot, mclust, poLCA

VignetteBuilder knitr

NeedsCompilation no

Maintainer Modesto Escobar <modesto@usal.es>

URL <https://modesto-escobar.github.io/netCoin/>

Encoding UTF-8

BugReports <https://github.com/Modesto-Escobar/netCoin/issues>

Acknowledgments Grants CSO2013-49278-EXP, PGC2018-093755-B100, PDC2022-133355-100, PID2023-147358NB-100 funded by MICIU/AEI/10.13039/501100011033 and by “European Union NextGenerationEU/PRTR”.

Config/pak/sysreqs libglpk-dev make libxml2-dev libx11-dev zlib1g-dev

Repository <https://modesto-escobar.r-universe.dev>

Date/Publication 2026-10-26 15:54:30 UTC

RemoteUrl <https://github.com/modesto-escobar/netcoin>

RemoteRef HEAD

RemoteSha 97ddaed6be0206425a6b8ca4a54d4f2932bc1256

Contents

netCoin-package	4
addAxes	6
addClusters	8
addCommunities	10
addDescription	12
addImage	13
addNetCoin	14
addTutorial	15
allNet	16
asGallery	18
asNodes	19
barCoin	20
calCentr	21
caring	23
caring_create_graphs	24
caring_read_file	25
clusterAgree	25
clusterMatch	27
cobCoin	29
coefTable	31
coexist	33
coin	35
coocur	36
d_netCorr	37
dice	39
dichotomize	40
distant	42
dyncohort	43
edgeList	44
ess	45
events	46
exhibit	47
expectedList	48
families	49
finches	50
fromIgraph	51
Galapagos	52
gallery	53

gallery3	55
get_panel_template	56
get_template	57
get_template2	58
glmCoin	59
glmCoin2	61
incTime	62
layoutCircle	63
layoutDist	64
layoutGrid	66
layoutsAlign	66
layoutsDist	69
links	71
logCoin	72
looking4clusters	74
lower	75
makeT	76
makeT2	76
marginsTable	77
mobileEdges	80
multigraphCreate	80
multiPages	82
netCoin	83
netCorr	86
netExhibit	88
netGallery	89
pathCoin	91
pieCoin	92
propCoin	93
renderLinks	94
replaceClusters	95
saveGhml	96
savePajek	97
shinyCoin	98
sim	99
sociologists	100
surCoin	101
surScat	103
timeCoin	112
toIgraph	114
works	114

netCoin-package

The netCoin package.

Description

netCoin combines the data analysis capabilities of R with the interactive visualization libraries of JavaScript to create **networks of coincidences, co-occurrences, correlations and regressions** that can be explored directly in a browser, embedded in HTML files or integrated into Shiny applications. It also generates **networked HTML galleries** and other **plots of coincidences**.

Details

Coincidence analysis detects what events, characters, objects, attributes, or characteristics tend to occur together within certain limits.

These given limits are call scenarios (S) and are considered to be the units of analysis, and as such they have to be placed in the rows of a matrix or data.frame.

In each i scenario, a series of J events X_j , which are to be represented as dichotomous variables X_j in columns, may occur (1) or may not occur (0). Scenarios and events constitute an incidence matrix (**I**).

Incidence matrix

	X_1	X_2	X_3	...	X_J
S_1	0	1	0	...	1
S_2	1	0	1	...	0
...	...	...	...	...	...
S_n	1	1	0	...	1

From this incidences matrix, a coincidence (**C**) matrix can be obtained with the function `coin`. In this matrix the main diagonal represents frequencies of X_j , while the others elements are number of coincidences between two events.

Coincidence matrix

	X_1	X_2	X_3	...	X_J
X_1	2	1	1	...	1
X_2	1	2	0	...	2
X_3	1	0	1	...	0
...	...	...	...	...	...
X_J	1	2	0	...	2

Once there is a coin object, a similarity matrix can be obtained. Similarity matrices available in netCoin are:

- Matching (m), Rogers & Tanimoto (t) Gower (g) Sneath (s) and Anderberg (and).

- Jaccard (j), dice (d), antiDice (a), Ochiai (o) and Kulczynski (k).
- Hamann (ham), Yule (y), Pearson (p), odds ratio (od) and Rusell (r).

Other measures that can be obtained from coin are:

- Relative frequencies (x), conditional frequencies (i) coincidence degree (cc) and probable degree of coincidence (cp).
- Haberman (h) and Z value of Haberman (z)

To obtain similarity and other measures matrices, the function `sim` elaborates a list of them.

Similarity matrix

	X_1	X_2	X_3	...	X_J
X_1	1.73	-.87	.87	...	-.87
X_2	-.87	1.73	-1.73	...	1.73
X_3	.87	-1.73	1.73	...	-1.73
...	...	...	...	...	...
X_J	-.87	1.73	-1.73	...	1.73

`edgeList` makes a collection of edges composed by a list of similarity measures whenever a criterion (generally $p(Z) < .50$) is met.

Edge list

	source	target	Haberman	P(z)
1	X1	X3	0.8660254	0.22509243
2	X2	X4	1.7320508	0.09084506

In order to make a graph, two data frames are needed: a nodes data frames with names and other nodes attributes (see `asNodes`) and an edge data frame (see `edgeList`). For more information go to [netCoin](#).

Author

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

- Escobar, M. (2009): "Redes Semanticas en Textos Periodisticos: Propuestas Tecnicas para su Representacion", en *Empiria*, 17, 13-39.
- Escobar, M.(2015): "Studying Coincidences with Network Analysis and Other Multivariate Tools", in *The Stata Journal*, 15(4), 1118-1156.
- Escobar, M. and J. Gomez Isla (2015): "The Expression of Identity through the Image: The Photographic Archives of Miguel de Unamuno and Joaquin Turina", en *Revista Espanola de Investigaciones Sociologicas*, 152, 23-46.
- Escobar, M. and L. Martínez-Urbe (2020): "Network Coincidence Analysis: The netCoin R Package", en *Journal of Statistical Software*, 93(11), 1-31.

addAxes	<i>Add axes to a netCoin object from surScat</i>
---------	--

Description

addAxes adds one or several planes of coordinates to a netCoin object from surScat, which the viewer then offers in a selector next to the graph.

Usage

```
addAxes(scatObj, axes, name=NULL, which=NULL, weight=NULL, axesLabels=NULL)
```

Arguments

scatObj	A 'netCoin' object.
axes	a matrix or data.frame with an even number of columns of coordinates, a pair per plane; a list of such matrices; or a 'looking4clusters' object, whose dimensionality reductions are taken as planes.
name	name of the new plane. When several planes are added at once, each one keeps its own name, and 'name' becomes a prefix.
which	reduction to take when 'axes' is a 'looking4clusters' object, given as a name or an index. When it is left empty, every reduction held by the object is added as its own plane.
weight	a vector of weights, one per case, used when collapsing cases into patterns. NULL (default) takes the weights surScat was called with, so that the plane being added is collapsed just as the one already drawn, and collapses by the plain arithmetic mean when there were none. A vector of ones asks for the plain mean of a scattergram that does have weights.
axesLabels	what to call the two axes once the scattergram holds more than one plane. NULL (default) lets every plane keep the titles of its own axes, which the viewer puts back whenever that plane is selected: the column names of an added plane, and the labels the first plane was drawn with, such as "F1 (23.5%)". A pair of strings puts the same titles on every plane instead, and NA keeps whatever the object already had.

Details

A scattergram drawn by surScat holds a single pair of coordinates, the one stated by its xaxis and yaxis arguments. addAxes adds further pairs to the same scattergram, so that the very same nodes, with their groups and their profiles, can be looked at from several projections without building a graph for each. The plane that was already drawn becomes the first entry of the selector, named after the axes it was drawn with.

Each pair of columns of axes makes up a plane: the first column is drawn horizontally and the second vertically. Planes are named after their two columns, as in "PC1-PC2", unless name states otherwise. A 'looking4clusters' object names them after its reductions ("pca", "tsne", "cmds",

"nmf", "umap", plus any added through its own `addreduction` function). The 'looking4clusters' package needs not be installed: the object is read through its list structure.

The coordinates must come from the same data, in the same order, as the one given to `surScat`: only their length is checked, so keeping the order is up to the caller. Coordinates given case by case are collapsed into patterns, by their (weighted) mean, when `surScat` was called with `vPatterns`. The weights are the ones `surScat` itself collapsed with, which it keeps, unless `weight` states otherwise: a plane added to a weighted scattergram is therefore comparable with the one already drawn without having to repeat the weights.

They must therefore hold one row per node, or one per case when the nodes are patterns. Anything else is an error, which is also what happens when `surScat` capped the nodes through its `maxN` argument: the node table then holds a random sample of the cases, or of the patterns, so coordinates computed on the whole data no longer line up with it and cannot be matched to it. Compute the coordinates on the node table of the object, which is that sample, or raise `maxN` so that every case is drawn.

The labels of the axes belong to the graph and not to each plane, so they keep describing the plane that `surScat` drew. Naming the added planes after their axes keeps the reader oriented.

Every clusterization the object already holds (see [surScat](#) and [addClusters](#)) gets, in `$clusters`, a further entry for the plane just added, named by joining the clusterization and the plane with a dot, as in "kmeans(3).petals", so that `$clusters` always holds one entry per clusterization-plane pair, covering the same planes as `$layouts`.

Value

A 'netCoin' object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
data(iris)
S <- surScat(iris, variables=names(iris)[1:4], type="pca", nclusters=3)

# a second plane from further principal components
P <- prcomp(iris[,1:4], scale.=TRUE)$x
S <- addAxes(S, P[,3:4])

# every reduction of a looking4clusters object
# S <- addAxes(S, l4cObject)
plot(S)

## End(Not run)
```

addClusters	<i>Add cluster columns to a netCoin object from surScat.</i>
-------------	--

Description

addClusters adds one or several cluster columns to a netCoin object from surScat.

Usage

```
addClusters(scatObj, clusters, name=NULL, sort=TRUE,
            weight=NULL, sourceData=NULL, which=NULL, maxGroups=25, rest=NULL)
```

Arguments

scatObj	A 'netCoin' object.
clusters	vector, factor, data.frame holding one clusterization per column, or clustering object (kmeans, poLCA, tidyLPA, looking4clusters).
name	name of the new cluster column. 'clusters' by default. When several clusterizations are added at once, each column keeps its own name, and 'name' becomes a prefix.
sort	sort clusters if requested (by mean coordinate on first layout axis).
weight	a vector of weights.
sourceData	optional data.frame for validation that clusters are in correct order.
which	clusterization to take when 'clusters' is a 'looking4clusters' object, given as a name or an index. Methods run for several numbers of groups yield their optimal solution, unless a specific number is asked for with the "name:k" form, as in which="kmeans:4". When it is left empty, every clusterization held by the object is added as its own column, one per number of groups, named after the method and that number, as in "kmeans(4)".
maxGroups	clusterizations with more groups than this are skipped with a warning, which keeps a continuous variable passed in by mistake from becoming a column of hundreds of groups. NULL adds them all.
rest	the group standing for a residual one, gathering what was left over rather than found. It is pinned after every real group instead of taking a place among them by sorting, and it is labelled with a word ("rest", "resto", "resta") where the others carry a number, so that a reader cannot take it for one more group of the series. addCommunities sets it; a clusterization with no residual group leaves it NULL.

Details

Each column is an ordered factor whose levels follow the group number. Their labels are prefixed with the name of the clusterization they come from, capitalized and without its trailing number of groups, so a column named "hclust(10)" is labelled "Hclust: 01" ... "Hclust: 10". Group numbers

are padded with zeros to the width of the largest one, which keeps them in numeric order wherever they are sorted as text.

Each clusterization is named after where it comes from and how many groups it holds, as in "kmeans(4)" for a method of a 'looking4clusters' object. The solutions of a 'tidyLPA' object are named "LPA(4)", and labelled "LPA: 1" ... "LPA: 4", since a latent profile analysis estimated for several numbers of profiles is one analysis. A 'tidyLPA' object estimated for several model specifications at once is no longer a single analysis, so its solutions carry the number tidyLPA gives each specification, as in "LPA1(4)" and "LPA6(4)".

Several clusterizations can be added in one call, either as the columns of a data.frame or as a 'looking4clusters' object. They must all come from the same data, in the same order, as the one given to surScat: only their length is checked, so keeping the order is up to the caller.

A clusterization must hold one assignment per node, or one per case when surScat collapsed the cases into patterns through vPatterns, in which case the cases of each pattern are reduced to their most frequent group. Anything else is an error, which is also what happens when surScat capped the nodes through its maxN argument: the node table then holds a random sample of the cases, or of the patterns, so a clusterization computed on the whole data no longer lines up with it and cannot be matched to it. Compute the clusters on the node table of the object, which is that sample, or raise maxN so that every case is drawn.

A 'looking4clusters' object usually holds many clusterizations at once: those added through its own addcluster function, plus one solution per number of groups for each method it has run. Leaving which empty brings all of them in, whereas which picks a single one. The 'looking4clusters' package needs not be installed: the object is read through its list structure.

Each clusterization added gets, in \$clusters, one entry per plane the object holds (see [surScat](#) and [addAxes](#)), named by joining the clusterization and the plane with a dot, as in scatObj\$clusters[["kmeans(3).petals"]]; each entry is a matrix with one row per group and its centroid coordinates. Where surScat collapsed the cases into patterns, a pattern spanning several groups keeps in its column the compound value (as in "Group: 1|Group: 2"), but there is still one centroid per group: the pattern takes part in the centroid of each group it spans with the (weighted) cases it holds of that group. These counts, one matrix per clusterization with a row per node and a column per group, are kept in the attribute "clusterShares" of the object, since the group of every case is no longer known once the cases are collapsed. \$clusterMeans is updated as well, with the mean of every group of every clusterization on every numeric variable of the node table (see [surScat](#)). Printing \$clusters, on its own or as part of the object, only states how many clusterizations and planes it covers.

Value

A 'netCoin' object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

addCommunities	<i>Add communities of cases to a netCoin object from surScat.</i>
----------------	---

Description

addCommunities adds one cluster column per community detection method to a netCoin object from surScat, taken from the network of correlations between its cases.

Usage

```
addCommunities(scatObj, data, variables=names(data), community="lo",
  scale=TRUE, method=c("pearson", "kendall", "spearman"),
  criteria="value", minL=0.95, maxL=Inf, pairwise=FALSE,
  minSize=0.01, maxCommunities=NULL, name=NULL, sort=TRUE,
  weight=NULL, maxGroups=25, maxNodes=NULL, seed=2020)
```

Arguments

scatObj	A 'netCoin' object returned by surScat .
data	the data.frame the scattergram was drawn from.
variables	the variables the cases are compared over. All of them by default.
community	one or several community detection methods, each given by its two-letter code or by its name: edge_betweenness ("ed"), fast_greedy ("fa"), label_prop ("la"), leading_eigen ("le"), louvain ("lo"), optimal ("op"), spinglass ("sp"), walktrap ("wa"). One cluster column is added per method.
scale	standardize each variable before transposing.
method	correlation coefficient: "pearson", "kendall" or "spearman".
criteria	which statistic the links are filtered on, "value" for the correlation itself or "p" for its significance.
minL	lowest value of criteria a pair of cases needs to be linked.
maxL	highest value of criteria a pair of cases may have to be linked.
pairwise	use pairwise complete observations for the correlations.
minSize	communities holding fewer cases than this are gathered into a residual group, read as a proportion of the cases when below 1 and as a count of them otherwise. Either reading is rounded up. Read as a proportion it never falls below two cases, since a community of a single case is not a group. 1 keeps every community apart.
maxCommunities	how many groups may come out at most, the residual one among them: 5 keeps the four largest communities apart and gathers every other into a fifth group. Nothing is gathered on this account when the groups already fit. NULL (default) leaves as many groups as clear minSize.
name	name of the new cluster column. Each column keeps the name of its method by default, and 'name' becomes a prefix when several methods are asked for at once.

sort	sort the groups by their mean coordinate on the first axis. The residual group is pinned last whatever its coordinate.
weight	a vector of weights for the mean coordinate the groups are sorted by. It does not weigh the correlations themselves.
maxGroups	clusterizations with more groups than this are skipped with a warning. See addClusters .
maxNodes	how large a network a method may be asked about, overriding the ceiling each one carries. NULL keeps those.
seed	the random seed each method is run from, as in surScat . NULL leaves the random state as it is. The random state of the caller is restored on exit either way.

Details

The cases are correlated with one another over the variables, which is the transposition of the usual reading of a data.frame: each case becomes a column and each variable a row, so that a link joins two cases whose profiles run parallel. The communities of that network are a grouping of cases, of the same shape as any other clusterization, and reach the node table through [addClusters](#), with the naming and the labelling described there. A louvain of five groups is a column named "louvain(5)", labelled "Louvain: 1" ... "Louvain: 5".

A correlation between two cases centres each of them on its own mean across the variables, so what it compares is the shape of their profiles. Standardizing the variables first puts them all on one footing; without it, the variable of widest range writes the profile on its own, which is why scale is on by default. Each correlation rests on as many points as there are variables, so comparing cases over very few of them says very little, and warns.

A network of cases leaves isolated ones behind, those reaching no other case above minL, and igraph returns each of them as a community of its own. minSize gathers those and any other small community into one residual group, which is pinned after the real ones and labelled with a word instead of a number, so that it reads apart from them in a legend. Without it a scattergram of three readable groups reports dozens and runs into maxGroups.

minSize sets a rule and lets the count of groups fall where it may; maxCommunities sets the count and lets the rule fall where it may, keeping the largest communities apart and gathering the rest, whatever their size. They work together, into one residual group, and a community falling to either of them ends up in it. Note that maxGroups is neither of these two: it gathers nothing, it merely refuses a clusterization of too many groups, so a maxCommunities at or below it means the column is always added.

Three of the methods draw on the random number generator, and so answer differently on the same network from one run to the next: louvain and label propagation sweep the nodes in a random order, and spinglass anneals from a random state. The other five (edge betweenness, fast greedy, leading eigenvector, optimal and walktrap) return the same communities every time. Left to run unseeded, the unstable three move their communities between runs, and with them the sizes minSize is read against, so that the same minSize gathers a different set of communities each time. seed therefore defaults to 2020, as in [surScat](#), and is set afresh before each method, so that a method answers the same whichever others were asked for alongside it in community.

The whole network is built before the links are filtered, so its cost grows with the square of the number of cases: some thousands of them are a matter of seconds and a few gigabytes, and well

beyond that it stops being feasible. The methods themselves do not all bear the same size either, and each one carries the ceiling it can meet: `cluster_optimal` is exponential and refuses above a hundred nodes, whereas `cluster_louvain` answers on thousands.

When `surScat` collapsed the cases into patterns through `vPatterns`, the communities are found case by case and then reduced to the most frequent one of each pattern, as any other clusterization is. When `surScat` capped the nodes through `maxN`, the network is built on the cases that were drawn, which the object records, so that the communities line up with the nodes that are actually plotted. The two together cannot be resolved: a sample of patterns cannot be traced back to the cases behind it, and that is an error.

Value

A 'netCoin' object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[addClusters](#), [surScat](#), [netCorr](#).

Examples

```
## Not run:
data <- data.frame(a=rnorm(300), b=rnorm(300), c=rnorm(300), d=rnorm(300))
S <- surScat(data, maxN=Inf)
S <- addCommunities(S, data, community=c("louvain", "walktrap"))
plot(S)

## End(Not run)
```

<code>addDescription</code>	<i>Adds a description to a 'netCoin' object.</i>
-----------------------------	--

Description

`addDescription` adds a description to a 'netCoin' object.

Usage

```
addDescription(x, description)
```

Arguments

<code>x</code>	A 'netCoin' object.
<code>description</code>	the description text.

Value

A 'netCoin' object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data(finches)
data(Galapagos)

Net<-allNet(Galapagos,nodes=finches, criteria="hyp", maxL=.05,
           lwidth ="Haberman",lweight="Haberman",
           size="frequency", color="species", layout="mds",
           main="Species coincidences in Galapagos Islands",
           note="Data source: Sanderson (2000)")

img <- system.file("extdata", "p.Crassirostris.png",
                  package="netCoin")
Net <- addDescription(Net,"Species coincidences in Galapagos Islands")

## Not run:
multi <- multigraphCreate(Network=Net)
multiPages(multi,"Graph description example",show=TRUE)

## End(Not run)
```

addImage	<i>Adds an image to a 'netCoin' object.</i>
----------	---

Description

addImage adds an image to a 'netCoin' object.

Usage

```
addImage(x, img)
```

Arguments

x	A 'netCoin' object.
img	character vector indicating the image path.

Value

A 'netCoin' object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data(finches)
data(Galapagos)

Net<-allNet(Galapagos,nodes=finches, criteria="hyp", maxL=.05,
           lwidth ="Haberman",lweight="Haberman",
           size="frequency", color="species", layout="mds",
           main="Species coincidences in Galapagos Islands",
           note="Data source: Sanderson (2000)")

img <- system.file("extdata", "p.Crassirostris.png",
                  package="netCoin")
Net <- addImage(Net,img)

## Not run:
multi <- multigraphCreate(Network=Net)
multiPages(multi,"Graph image example",show=TRUE)

## End(Not run)
```

addNetCoin

Networked coincidences.

Description

addNetCoin changes netCoin object attributes.

Usage

```
addNetCoin(x, ...)
```

Arguments

x	A 'netCoin' object.
...	Any netCoin argument.

Details

The object is rebuilt out of its own options, with the ones stated here replacing them, so anything not among the arguments of [netCoin](#) would be lost in the rebuilding and is carried over instead: the planes [addAxes](#) added, and what a scattergram records about how its nodes stand for the cases, which [addClusters](#), [addCommunities](#) and [replaceClusters](#) read afterwards. Stating layout replaces the planes, as asked; replacing nodes with a table of another length carries none of it over, since none of it would describe those nodes.

Nodes drawn with no label at all, as [surScat](#) draws them, keep it that way. A label is restored by naming its column, as in `label="name"`.

Value

This function returns a netCoin object. If the 'dir' attribute is specified, the function creates a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C) # edge data frame
net <- netCoin(N, E) # netCoin object
net <- addNetCoin(net, color="frequency") # netCoin object with color applied to nodes
```

addTutorial

Adds a tutorial for the gallery.

Description

addTutorial adds a tutorial for a gallery.

Usage

```
addTutorial(x, image = NULL, description = NULL, help = FALSE)
```

Arguments

x	object of class gallery_rd3.
image	character vector indicating the image path, header for the tutorial.
description	a character string indicating a description text to insert in the tutorial.
help	Should you press the help button to show the tutorial? ('FALSE' by default)

Value

Object of class gallery_rd3.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data("finches")
finches$species <- system.file("extdata", finches$species,
  package="netCoin") # copy path to the species field
g <- gallery(finches, image="species", main="Species in Galapagos Islands",
  note="Data source: Sanderson (2000)")

g <- addTutorial(g,
  description="Here you can see different finches species in Galapagos islands.")

## Not run:
plot(g)

## End(Not run)
```

allNet

Networked coincidences from incidences data.

Description

allNet produces a network object of coincidences from a data frame or a matrix with dichotomous values.

Usage

```
allNet(incidences, weight = NULL, subsample = FALSE, pairwise = FALSE,
  minimum=1, maximum = nrow(incidences),
  sort = FALSE, decreasing = TRUE,
  frequency = FALSE, percentages = TRUE,
  procedures = "Haberman", criteria = "Z", Bonferroni = FALSE,
  support = -Inf, minL = -Inf, maxL = Inf,
  directed = FALSE, diagonal = FALSE,
  sortL = NULL, decreasingL = TRUE,
  igraph = FALSE, dir=NULL, ...)
```

Arguments

incidences	an incidence matrix or data frame with only 0/1 variables.
weight	a vector of weights. Optimal for data.frame tables.
subsample	retract the analysis to scenarios with at least one event.
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.
minimum	minimum frequency to be considered.

maximum	maximum frequency to be considered.
sort	sort the coincidence matrix according to frequency of events.
decreasing	decreasing or increasing sort of the matrix.
frequency	a logical value true if frequencies are to be shown. Default = FALSE.
percentages	a logical value true if percentages are to be shown. Default = TRUE.
procedures	a vector of statistics of similarity. See below.
criteria	statistic to be use for selection criteria.
Bonferroni	Bonferroni criterium of the signification test.
support	minimum value of the frequency of the coincidence to be edged.
minL	minimum value of the statistic to include the edge in the list.
maxL	maximum value of the statistic to include the edge in the list.
directed	includes same edges only once.
diagonal	includes auto-links.
sortL	sort the list according to the values of a statistic. See below.
decreasingL	order in a decreasing way.
igraph	Produces an igraph object instead of a netCoin object if TRUE.
dir	a "character" string representing the directory where the web files will be saved.
...	Any netCoin argument.

Details

Possible measures in procedures are

- Frequencies (f), Relative frequencies (x), Conditional frequencies (i), Coincidence degree (cc), Probable degree (cp),
- Expected (e), Confidence interval (con)
- Matching (m), Rogers & Tanimoto (t), Gower (g), Sneath (s), Anderberg (and),
- Jaccard (j), Dice (d), antiDice (a), Ochiai (o), Kulczynski (k),
- Hamann (ham), Yule (y), Pearson (p), odds ratio (od), Rusell (r),
- Haberman (h), Z value of Haberman (z),
- Hypergeometric p greater value (hyp).
- Convert a matrix into an edge list (shape).

Value

This function creates a netCoin object (or igraph) and, if stated, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
allNet(data) # network object
```

asGallery

Images in a grid gallery.

Description

asGallery produces a gallery_rd3 object.

Usage

```
asGallery(net)
```

Arguments

net is a netCoin object. See [netCoin](#)

Value

Object of class gallery_rd3.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data("Galapagos")
data("finches")
finches$species <- system.file("extdata", finches$species,
                               package="netCoin") # copy path to the species field
Net <- allNet(Galapagos, nodes=finches, criteria="hyp", maxL=.05,
             image="species",
             main="Species coincidences in Galapagos Islands",
             note="Data source: Sanderson (2000)")
gallery <- asGallery(Net)
```

asNodes	<i>Nodes data frame.</i>
---------	--------------------------

Description

Nodes data frame from either an edge list or a coin object.

Usage

```
asNodes(C, frequency = TRUE, percentages = FALSE, language = c("en", "es", "ca"))
```

Arguments

C	has to be an edge list or, better, a coin object.
frequency	add frequency of nodes
percentages	add nodes percentages
language	a character vector (es=spanish; en=english; ca=catalan).

Value

A data frame with nodes' names and their frequency and/or percentages if the input is a coin object

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# From a random incidence matrix I(25X4)
I <- matrix(rbinom(100, 1, .5), nrow = 25, ncol = 4,
  dimnames = list(NULL, c("A", "B", "C", "D")))
C <- coin(I)
asNodes(C)
```

barCoin	<i>Networked coincidences.</i>
---------	--------------------------------

Description

barCoin produces a barCoin object.

Usage

```
barCoin(data, variables = colnames(data), commonlabel = NULL,
        dichotomies = c("_all", "_none"), valueDicho = 1, weight = NULL,
        subsample = FALSE, sort = NULL, decreasing = TRUE, nodes = NULL,
        name = NULL, select = NULL, scalebar = FALSE, note = NULL,
        label = NULL, text = NULL, color = NULL, defaultColor = "#1f77b4",
        expected = FALSE, confidence = FALSE, level = .95, significance = FALSE,
        minimum = 1, maximum = nrow(data), percentages = FALSE,
        criteria = c("Z", "hyp"), Bonferroni = FALSE,
        support = 1, minL = -Inf, maxL = 1,
        language = c("en", "es", "ca"), cex = 1.0, dir = NULL)
```

Arguments

data	a data frame
variables	a vector of variables included in the previous data frame
commonlabel	a vector of variables whose names are to be included in nodes labels
dichotomies	a vector of dichotomous variables to appear as just one categorie
valueDicho	value to be selected for dichotomous variables. Default is 1
weight	a vector of weights. Optimal for data.frame tables.
subsample	retrict the analysis to scenarios with at least one event.
sort	name of the vector in the nodes data frame to order the graph.
decreasing	decreasing or increasing sort of the graph order.
nodes	a data frame with at least two vectors of names and incidences.
name	name of the vector with names in the nodes data frame.
select	Name of the event (in nodes name column) to start the visualization.
scalebar	Should the bars fill the screen height? Default = FALSE.
note	lower title of the graph.
label	name of the vector with labels in the nodes data frame.
text	name of the vector with html text in the nodes data frame.
color	name of the vector with color variable in the nodes data frame.
defaultColor	a character vector giving a valid html color.
expected	name of the vector with expected coincidences in the links data frame.

confidence	name of the vector with confidence interval in the links data frame.
level	confidence level
significance	name of the vector with significance in the links data frame.
minimum	minimum frequency to be considered.
maximum	maximum frequency to be considered.
percentages	a logical value true if percentages are to be shown. Default = TRUE.
criteria	statistic to be use for selection criteria.
Bonferroni	Bonferroni criterium of the signification test.
support	minimum value of the frequency of the coincidence to be edged.
minL	minimum value of the statistic to include the edge in the list.
maxL	maximum value of the statistic to include the edge in the list.
language	a character vector (es=spanish; en=english; ca=catalan).
cex	number indicating the amount by which plotting text should be scaled relative to the default. Default = 1.
dir	a "character" string representing the directory where the web files will be saved.

Value

Object of class barCoin.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]

barCoin(data,dichotomies="_all")
barCoin(data,dichotomies="_all",confidence=TRUE,percentages=TRUE)
```

calCentr

Categorize a network

Description

This function calculates the centrality measures of a network.

Usage

```
calCentr(graph,
  measures = c("degree", "wdegree", "closeness", "betweenness", "eigen"),
  order = "")
```

Arguments

graph	A netCoin object.
measures	Character vector of the measures to be calculated (See details).
order	Sort the data.frame by the different measures.

Details

This function reproduces some of the most significant classic Social Network Theory's centrality measures. See Wasserman (1994), Freeman (1978), or Bonacich & Lloyd (2001) to know more.

- a) Degree = Degree centrality is measured by the total amount of direct links with the other nodes.
- b) Closeness = Closeness centrality is meant to measure one node to the others nodes' sum distances
- c) Betweenness = Betweenness centrality measures one node undertaking "mediation" role in a network.
- d) Eigen = Eigenvector centrality measures a node's importance while giving consideration to the importance of its neighbors.

By default, measures = "all", thus all the measures will be calculated. The function can be applied to an [igraph](#) or a [netCoin](#) object. In case the graph is undirected, it will show the degree, weighted degree, closeness, betweenness and eigen degree. Moreover, if it is directed, it will show the indegree, windegree and outdegree, both weighted and unweighted (See example.).

Value

This function creates a list containing two elements: 1) a data.frame (nodes) with all the centrality measures applied to the graph and 2) another data.frame (graph) with this measures applied to the whole network.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C) # edge data frame
G <- netCoin(N, E, showArrows=TRUE) # netCoin object
calCentr(G, "all")
```

caring	<i>Produce interactive multi graphs.</i>
--------	--

Description

caring produce an interactive multi graph from caring arguments.

Usage

```
caring(filepath,arguments)
```

Arguments

filepath	The path to the data file.
arguments	a list with the caring exported arguments.

Value

This function returns a mGraph object. The function creates a folder in your computer with an HTML document named index.html which contains the graph. This file can be directly opened with your browser.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
filename <- "demo.sav"
arguments <- list(
  variables = c("Gender", "Income category in thousands",
    "Primary vehicle price category"),
  plot = "network"
)
graph <- caring(filename,arguments)
plot(graph)

## End(Not run)
```

caring_create_graphs *Produce interactive multi graphs.*

Description

caring_create_graphs produce an interactive multi graph from caring arguments.

Usage

```
caring_create_graphs(data,arguments)
```

Arguments

data	A data frame.
arguments	a list with the caring exported arguments.

Value

This function returns a mGraph object. The function creates a folder in your computer with an HTML document named index.html which contains the graph. This file can be directly opened with your browser.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data <- data.frame(Gender=c(rep("Man",3),rep("Woman",3)),
                  Opinion=c("Yes","Yes","No","No","No","Yes"))
arguments <- list(
  variables = c("Gender", "Opinion"),
  dichotomies = "Opinion",
  valueDicho = "Yes",
  plot = c("surCoin","network")
)
graph <- caring_create_graphs(data,arguments)
## Not run:
plot(graph)

## End(Not run)
```

caring_read_file	<i>Read sav, dta, xlsx, tsv and csv files.</i>
------------------	--

Description

caring_read_file reads sav, dta, xlsx, tsv and csv files and prepares data for caring.

Usage

```
caring_read_file(filepath)
```

Arguments

filepath	The path to the data file.
----------	----------------------------

Value

This function returns a data frame.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:  
filename <- "demo.sav"  
data <- caring_read_file(filename)  
  
## End(Not run)
```

clusterAgree	<i>Agreement between the classifications of a scattergram.</i>
--------------	--

Description

clusterAgree measures the agreement between a model classification and the other classifications held by a netCoin object from surScat through ARI, NMI and VI, which match no categories and take therefore classifications with any number of groups.

Usage

```
clusterAgree(x, model, columns=NULL, weights=NULL, modal=TRUE, sort=NULL, digits=3)
```

Arguments

<code>x</code>	a 'netCoin' object from <code>surScat</code> , or the <code>data.frame</code> of its nodes.
<code>model</code>	the classification everything else is compared with: the name of a node column, or a vector holding one value per node. Where the nodes are patterns, a vector holding one value per case is taken as well, and the comparison then goes case by case.
<code>columns</code>	the classifications to compare. <code>NULL</code> (default) takes those <code>surScat</code> recorded in the "clusterColumns" attribute and, where there are none, every factor or character column of the node table.
<code>weights</code>	what every unit counts for. The name of a numeric node column, or a vector holding one value per node or per case. <code>NULL</code> (default) weighs every node by the cases it stands for when the nodes are patterns, and counts one per unit otherwise. <code>weights=1</code> forces the plain count of nodes.
<code>modal</code>	where the nodes are patterns, whether to keep the modal group of each pattern. See clusterMatch .
<code>sort</code>	the column to sort the result by, from the largest down; a vector of names sorts by all of them at once. The prefix "-" turns one column around. <code>NULL</code> (default) keeps the order the classifications came in.
<code>digits</code>	decimal places the numeric columns are rounded to.

Details

Splitting finer never harms precision — in the limit, one group per case leaves it at 1 — so the measures of a confusion matrix cannot arbitrate between classifications with different numbers of groups, and [clusterMatch](#) refuses to compare them at all. These three indices are there for that: they read the two classifications as partitions and need no matching of categories.

ARI, the adjusted Rand index, walks through the pairs of cases, counting whether the two classifications put them together or keep them apart, and corrects for chance: 0 is what independent partitions yield, 1 an exact match. It may go slightly negative.

NMI, the normalised mutual information, tells how much one classification cuts down the uncertainty about the other, from 0 to 1. It is not corrected for chance, so it tends to rise with the number of categories and does not arbitrate well between solutions of different size.

VI, the variation of information, is a distance in nats: 0 is an exact match and it grows with the discrepancy. `VI.n` divides it by $\log(N)$ to keep it between 0 and 1. Note when sorting that in these two the small value is the good one, so `sort="-VI"` is usually what is wanted.

Where the nodes are patterns, weighting and modal groups work just as in [clusterMatch](#). Weights are taken to be on a scale of cases, which is what the default weighting yields: the count of pairs behind ARI is extended continuously, and weights given as proportions would make it meaningless.

Value

A `data.frame` with one row per classification, sorted as `sort` states. It carries as attributes the contingency table of every classification ("tables"), the number of categories of the model ("kModel"), the unit of analysis ("unit"), the weights actually applied ("weights") and, where there were any, the patterns spanning several groups ("spanning").

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[clusterMatch](#), [surScat](#), [addClusters](#)

Examples

```
## Not run:
data(iris)
S <- surScat(iris, variables=names(iris)[1:4], type="pca", nclusters=2:4)

# every classification, whatever its number of groups
A <- clusterAgree(S, "Groups(3)", sort="ARI")
A
attr(A, "tables")[[1]]

# the most concordant on top, VI being a distance
clusterAgree(S, "Groups(3)", sort="-VI")

## End(Not run)
```

clusterMatch

Compare the classifications of a scattergram, category by category.

Description

clusterMatch compares a model classification with the other classifications held by a netCoin object from surScat, matching their categories one to one and reporting sensitivity, precision, F1 and specificity.

Usage

```
clusterMatch(x, model, columns=NULL, weights=NULL, align=TRUE, weighted=TRUE,
             modal=TRUE, sort=NULL, digits=3)
```

Arguments

x	a 'netCoin' object from surScat, or the data.frame of its nodes.
model	the classification everything else is compared with: the name of a node column, or a vector holding one value per node. Where the nodes are patterns, a vector holding one value per case is taken as well, and the comparison then goes case by case.
columns	the classifications to compare. NULL (default) takes those surScat recorded in the "clusterColumns" attribute and, where there are none, every factor or character column of the node table.

weights	what every unit counts for. The name of a numeric node column, or a vector holding one value per node or per case. NULL (default) weighs every node by the cases it stands for when the nodes are patterns, and counts one per unit otherwise. weights=1 forces the plain count of nodes.
align	whether to realign the categories of every classification with those of the model before comparing them. TRUE by default; FALSE compares them as they come, which only makes sense where the categories already mean the same thing in both.
weighted	whether to add, next to the plain means of F1 and specificity, the means weighted by the size of every category (.w).
modal	where the nodes are patterns, whether to keep the modal group of each pattern. See the details.
sort	the column to sort the result by, from the largest down; a vector of names sorts by all of them at once, the first as the main criterion. The prefix "-" turns one column around: c("-k", "F1"). NULL (default) keeps the order the classifications came in.
digits	decimal places the numeric columns are rounded to.

Details

The groups of a classification carry no name of their own: group 1 of one method need not be group 1 of another, as the numbering is arbitrary. Before comparing, the categories of every classification are therefore realigned with those of the model through the optimal assignment (Hungarian algorithm), which maximizes the diagonal of the contingency table. That demands a one-to-one correspondence, and `clusterMatch` compares therefore only the classifications holding as many categories as the model: without a bijection these measures are not defined. `clusterAgree` is there for the rest.

The measures follow the definitions of `caret::confusionMatrix` (one against the rest, prediction down the rows and reference along the columns) but are computed here so as not to depend on `caret`. Every one of them is averaged over the categories in two ways: the plain mean treats them alike, and a small group badly classified sinks it; the one weighted by the size of each group measures what happens to a case drawn at random. Sensitivity is given no weighted mean, as it would coincide with Accuracy.

Where `surScat` has collapsed the cases into patterns, every node stands for several cases and counting it once would give the rare pattern the same say as the common one. The nodes are therefore weighed by the cases they stand for (the "caseToPattern" attribute, and "caseWeight" where `surScat` was given weights), so that the tables come out in cases.

Pattern nodes bring a limitation of the object itself: `surScat` keeps in every cluster column all the groups the pattern spans, as text ordered from the commonest down ("Group: 2|Group: 1"), and loses which group each case went to. The comparison is then made with the modal group of every pattern, and how many patterns span more than one is reported: where they are many, the figures are optimistic and it is worth comparing case by case, passing the model as a vector of cases and classifications that have not been collapsed.

Value

A data.frame with one row per classification, sorted as sort states. It carries as attributes the confusion matrix of every classification ("tables"), its measures category by category ("byClass"), the matching of categories ("mapping"), the unit of analysis ("unit"), the weights actually applied ("weights") and, where there were any, the patterns spanning several groups ("spanning").

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[clusterAgree](#), [surScat](#), [addClusters](#)

Examples

```
## Not run:
data(iris)
S <- surScat(iris, variables=names(iris)[1:4], type="pca", nclusters=2:4, clusters="hclust")

R <- clusterMatch(S, "Groups(3)")
R
attr(R, "byClass")[[1]] # measures category by category
attr(R, "tables")[[1]] # the realigned confusion matrix
attr(R, "mapping")[[1]] # which category was matched with which

# sorted by F1, and by the simplest classifications first
clusterMatch(S, "Groups(3)", sort="F1")
clusterMatch(S, "Groups(3)", sort=c("-k", "F1"))

# with an outside vector rather than a node column
clusterMatch(S, iris$Species)

## End(Not run)
```

cobCoin

CobWeb graph (Upton).

Description

cobCoin produces a netCoin object to graph a CoWeb graphic (Upton 2000).

Usage

```
cobCoin(data, variables=names(data), degree=0, significance=.05, ...)
```

Arguments

data	a data frame.
variables	a vector of variables included in the previous data frame.
degree	degree to rotate the categories of the graph
significance	p value maximum to represent the links
...	Any surCoin or netCoin argument.

Value

This function creates a netCoin object (or igraph) and, if stated, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Upton, Graham J.G. (2000) Cobweb diagrams for multiway contingency tables. *The Statistician*, **49**(1), 79-85.

Examples

```
# A data frame with two variables Gender and Opinion
frame <- data.frame(Gender=c(rep("Man",3),rep("Woman",3)),
                    Opinion=c("Yes","Yes","No","No","No","Yes"))
cobCoin(frame, significance=.5, degree=45) # netCoin object

# A data frame with two variables (Gender and Hand) and nodes
input <- data.frame(
  Gender = c("Women", "Men", "Men", "Women", "Women","Men",
            "Men", "Men", "Women", "Women", "Men", "Women"),
  Hand   = c("Right", "Left","Right", "Right", "Right", "Right",
            "Left", "Right", "Right", "Left","Right", "Right"))
nodes <- data.frame(
  name   = c("Gender:Men","Gender:Women", "Hand:Left", "Hand:Right"),
  label  = c("Women(50\u25b2)","Men(50\u25b2)",
            "Left hand(25\u25b2)", "Right hand(75\u25b2)"))
G <- cobCoin(input, nodes=nodes, label="label", degree=22.5,
             showArrows=TRUE, significance=1)
```

coefTable

*Coefficient Table from Regression Graphs***Description**

Builds a side-by-side coefficient table from one or several objects returned by [glmCoin](#). It reads the `coeffs` list stored in them (one data frame of coefficients per equation) and the equations data frame (`n` and pseudo-R2 of each model). Note that it does not read `links`, which holds average marginal effects rather than coefficients: that is what [marginsTable](#) tabulates. Each column block corresponds to one combination of object, equation and Target, so the same function covers hierarchical regressions (one response, nested sets of predictors), several responses in a single object, multinomial responses (one block per category) and any combination of these.

Usage

```
coefTable(G, labels = NULL, columns = c("coef", "sig", "se"),
          stfactors = c("nostandard", "x", "long"),
          blocks = c("entry", "variable", "none"), intercept = FALSE,
          sig = c(0.05, 0.01, 0.001), digits = 3, digits.se = digits,
          footer = c("n_obs", "r2", "dr2"), note = FALSE,
          file = NULL, format = NULL)
```

Arguments

- | | |
|-----------|--|
| G | A netCoin object created by glmCoin , or a list of them. A named list uses its names as column labels. Objects built with earlier versions have no <code>coeffs</code> and must be rebuilt. |
| labels | Optional character vector of column-block labels, one per model/target combination. If missing, labels are built automatically from the components that actually vary (model, response and, for multinomial responses, category), so that no redundant text is shown. |
| columns | Which columns form the block repeated for each model, shown in the order given: "coef" (estimate), "se" (standard error in parentheses), "sig" (significance stars), "sta" (standardized coefficient, see <code>stfactors</code>), "exp" (exponentiated coefficient), "ci" (confidence interval) and "p" (p value). At least one is required. The "exp" column is only filled for equations whose link is log (rate ratios) or logit (odds ratios); it is deliberately left empty for other links, so that a probit coefficient is never exponentiated. |
| stfactors | Convention used to standardize the coefficients shown in the "sta" column. In every case the estimate is multiplied by the weighted standard deviation of its term and divided by the scale of the response: $sd(y)$ when the response is on its own scale (gaussian models), $sd(y^*)$ of the underlying latent variable when <code>stfactors="long"</code> and the link is logit or probit (Long's convention), and 1 otherwise. The three values differ in the treatment of qualitative terms: "nostandard" (default) leaves the dummies in their original metric, so their coefficient keeps reading as the effect of belonging to the category; "x" standardizes them |

	as well, by the standard deviation of the dummy; and "long" standardizes them and divides by the latent scale.
blocks	How predictor rows are grouped. "entry" (default) groups them by the equation in which each predictor first appears, which reproduces the theoretical blocks of a hierarchical regression; "variable" groups the categories of each explanatory variable, whose name heads the block; "none" shows a flat list. A named list of predictor names may also be given to define the groups explicitly.
intercept	If TRUE, the constant of each equation is shown in a last row, after the predictor blocks. It is FALSE by default because the constant does not belong to any of them.
sig	Numeric vector with the significance thresholds of the stars, sorted internally in decreasing order: one star for p below the largest threshold, two for the next, and so on (any number of thresholds is allowed, e.g. <code>sig=c(.1, .05, .01, .001)</code> for a four-star scale).
digits	Number of decimals for the coefficients.
digits.se	Number of decimals for standard errors and confidence limits. Defaults to digits.
footer	Character vector with the statistics to show as bottom rows. Besides any column of G\$equations (n, n_obs, r2, r2.mcfadden, r2.nagelkerke, logLik, aic, bic, deviance, null.deviance, df.residual), three are computed here: k (number of predictors), dr2 (increment in R2 over the reference model) and ref (the reference model itself). The reference is found automatically as the largest model strictly nested in each equation and fitted on the same response. Categories of a multinomial response share a single equation, so its statistics are shown once, under the first of its blocks. Use NULL to omit the footer.
note	If TRUE, a legend with the meaning of the stars (built from sig) and of the standardization in use is added below the table, both when printing and in the exported formats.
file	Optional file name to export the table. The format is inferred from the extension (.txt, .tex or .csv) unless format is given.
format	Export format: "text", "latex" or "csv" (the latter opens directly in Excel). If file is missing, a default name is used.

Details

Rows are the union of the predictors of all the equations, in order of first appearance; an empty cell means that the predictor was not included in that equation, which is not the same as a zero coefficient.

Because coeffs and equations keep the same order, the columns are matched to their fit statistics by position, without comparing model strings.

Because R2 and its increments are only comparable when the equations are fitted on the same cases, a warning is issued whenever the models differ in their number of observations, and the fact is recorded in the note when note=TRUE. This typically happens when a predictor added in a later step has missing values.

Value

A character matrix of class `coefTable`, printed with grouped headers (one block of columns per model, response or category). The bottom rows show the statistics selected in footer. When exporting, the table is returned invisibly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[glmCoin](#), [marginsTable](#)

Examples

```
data(birthwt, package="MASS")
birthwt$smoke <- factor(birthwt$smoke, labels=c("no","yes"))
birthwt$race  <- factor(birthwt$race,  labels=c("white","black","other"))

## Hierarchical regression: one response, nested sets of predictors.
## Blocks are deduced from the equation in which each predictor enters.
m1 <- glmCoin("bwt ~ age + lwt", birthwt)
m2 <- glmCoin("bwt ~ age + lwt + smoke", birthwt)
m3 <- glmCoin("bwt ~ age + lwt + smoke + race", birthwt)
coefTable(list(m1, m2, m3), note=TRUE)

## Standardized coefficients: quantitative terms only, then all of them
coefTable(m3, columns=c("coef","sta"))
coefTable(m3, columns=c("coef","sta"), stfactors="x")

## Confidence intervals instead of standard errors, with fewer decimals
coefTable(list(m1, m2, m3), columns=c("coef","sig","ci"), digits=2, digits.se=1)

## Logistic regression: odds ratios beside the coefficients
g <- glmCoin(c("bwt ~ age + lwt", "low ~ age + lwt, binomial"), birthwt)
coefTable(g, columns=c("coef","sig","exp"), footer=c("n_obs","r2"))

## Multinomial response: one block of columns per category
data(housing, package="MASS")
h <- glmCoin("Sat ~ Infl + Type + Cont, multinomial", housing, weights="Freq")
coefTable(h, blocks="variable", columns=c("coef","sig"))
```

Description

`coexist` produces interactive graphs representing coexistence. Two periods or lifes coexist if they share a given number of years.

Usage

```
coexist(periods, name="name", start="start", end="end",
        fields=names(periods), plusstart=0, minusend=0, igraph=FALSE, ...)
```

Arguments

periods	a data frame with at least three vectors with name, start and end of the periods.
name	name of the vector with names in the periods data frame.
start	name of the vector with starts in the periods data frame.
end	name of the vector with ends in the periods data frame.
fields	vector of the names of the periods data frame to be taken into account.
plusstart	number of years to be trimmed at the beginning of each period.
minusend	number of years to be trimmed at the end of each period.
igraph	produces an igraph object instead of a netCoin class.
...	Any netCoin argument.

Details

Two periods coexists if they have at least one year in common. Periods can be trimmed at the beginning or at the end.

Value

This function creates a netCoin object (or igraph) and, if plotted, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

A netCoin object has three elements:

nodes	A data frame with the periods.
links	A data frame with the events.
options	A list of options for the interactive graph.

Note

Periods could be the life of people, in whose case start is their birth and end their death year.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[timeCoin](#) and [dyncohort](#)

Examples

```
# With sociologists data.
data(sociologists)
# Produce an igraph object:
coexist(sociologists, start="birth", end="death", igraph=TRUE)
# Omit as coexistence the 15 first years of life:
sociologists$picture <- system.file("extdata", sociologists$picture,
                                     package="netCoin") # copy path to the picture field
coexist(sociologists, start="birth", end="death", plusstart=15, image="picture", imageName="name")
```

coin

Coincidence matrix.

Description

A coin object is a squared matrix of coincidences with its names and the frequency of events as attributes.

Usage

```
coin(incidences, minimum = 1, maximum = nrow(incidences),
     sort = FALSE, decreasing = TRUE,
     total = FALSE, subsample = FALSE,
     weight = NULL, pairwise = FALSE)
```

Arguments

incidences	an incidence matrix or data frame with only 0/1 variables
minimum	minimum frequency to be considered
maximum	maximum frequency to be considered
sort	sort the coincidence matrix according to frequency of events
decreasing	decreasing or increasing sort of the matrix
total	add one first row and column with total
subsample	retract the analysis to scenarios with at least one event
weight	a vector of weights. Optimal for data.frame tables
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.

Details

Produce a matrix of coincidences from a matrix of incidences.

Value

An object of coin class

n	Number of scenarios (rows of the incidence matrix)
f	Coincidence matrix

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
## Random incidence matrix: 25 scenarios, 4 events.
I <- matrix(rbinom(100, 1, .5), nrow = 25, ncol = 4,
            dimnames = list(NULL, c("A", "B", "C", "D")))
coin(I, sort = TRUE)

## Hair by Eye by Sex table from M. Friendly (2000)
data(HairEyeColor)
H<-as.data.frame(HairEyeColor)
W<-H$Freq
I<-dichotomize(H,c("Hair","Eye","Sex"),add=FALSE)
coin(I,w=W)
```

coocur

Cooccurrence matrix.

Description

A cooccurrence object consists of a matrix with the number of occurrences in its main diagonal and the number of cooccurrences outside this diagonal. Besides, this object has two attributes: 1) n is the total of the sum of the occurrences in each row. 2) m is the sum of the maximum number of occurrences in each row.

Usage

```
coocur(occurrences, minimum = 1, maximum = Inf,
       sort = FALSE, decreasing = TRUE)
```

Arguments

occurrences	an occurrence matrix or data frame
minimum	minimum frequency to be considered
maximum	maximum frequency to be considered
sort	sort the coincidence matrix according to frequency of events
decreasing	decreasing or increasing sort of the matrix

Details

Produce a matrix of cooccurrences from a matrix of occurrences.

Value

An object of cooc class with a cooccurrence matrix. It has two attributes:

n	Total sum of occurrences)
m	Sum of maximum occurrences in each row of the occurrence matrix

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Tossing two coins five times.
D<-data.frame(Head=c(2,1,1,0,2),Tail=c(0,1,1,2,0))
coocur(D)
```

d_netCorr	<i>Dynamic networked correlations.</i>
-----------	--

Description

netCorr produces a network object of dynamic correlations. Its input has to be at least one set of quantitative variables.

Usage

```
d_netCorr(variables, nodes = NULL, weight=NULL,
           pairwise=FALSE, minimum=-Inf, maximum=Inf,
           frequency=FALSE, means=TRUE,
           method=c("pearson", "kendall", "spearman"), criteria="value", Bonferroni=FALSE,
           minL=0, maxL=Inf,
           sortL=NULL, decreasingL=TRUE,
           factorial=c("null", "pc", "nf", "vf", "of"),
           components=TRUE, backcomponents=FALSE,
           sequence=seq(.20, 1, .01), textFilter=c(1, .99), speed=50,
           dir=NULL, ...)
```

Arguments

variables	a data frame with at least two quantitative variables.
nodes	a data frame with at least one vector of names and other information from the nodes
weight	a vector of weights. Optimal for data.framed tables
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.
minimum	minimum mean to be considered
maximum	maximum mean to be considered

frequency	a logical value true if frequencies are to be shown. Default=FALSE.
means	a logical value true if means are to be shown. Default=TRUE.
method	a vector of statistics of similarity. Pearson correlation by default. Spearman and Kendall are also possible
criteria	statistic to be use for selection criteria.
Bonferroni	Bonferroni criterium of the signification test.
minL	minimum value of the statistic to include the edge in the list.
maxL	maximum value of the statistic to include the edge in the list.
sortL	sort the list according to the values of a statistic. See below
decreasingL	order in a decreasing way.
factorial	factorial layout: Principal components (pc), factorial (nf), factorial with varimax rotation (vf), and factorial with oblimin rotation (of) are possible
components	display following graph in sequence only when the components are unequal
backcomponents	display following graph in sequence even if there are less components
sequence	evolution of the dinamic graphs. c(first threshold, last threshold, step)
textFilter	limits for showing the correlations as text on the graph. This limit is the minimum value of the first (absolute), and the second (threshold plus its value)
speed	speed of the dynamic evolution from 0 to 100
dir	a "character" string representing the directory where the web files will be saved.
...	any netCoin argument.

Value

The function creates a list object to create a multigraph and eventually a folder in the computer with an HTML document named index.html which contains the produced dynamic graphs. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
data(iris)
d_netCorr(iris[,1:4],ltext="value", sequence=seq(.80, 1, .05), zoom=3, textFilter=.85,
  main="Correlations between measurements of Iris Species", components=FALSE,
  note="Anderson, Edgar (1935) y Fisher, R. A. (1936)") # network object
```

`dice`*Data: Roll a die (100 times).*

Description

Data frame with events as result.

Usage

```
data("dice")
```

Format

A data frame with 100 observations (scenarios) on the following 11 variables (events):

`dice` : a numeric vector, representing dice results

`1` : a dichotomous vector of the elemental event "1"

`2` : a dichotomous vector of the elemental event "2"

`3` : a dichotomous vector of the elemental event "3"

`4` : a dichotomous vector of the elemental event "4"

`5` : a dichotomous vector of the elemental event "5"

`6` : a dichotomous vector of the elemental event "6"

`odd` : a dichotomous vector of odd events

`even` : a dichotomous vector of even events

`small` : a dichotomous vector of small number events

`large` : a dichotomous vector of large number events

Source

Random extraction via `sample(1:6,100,replace=TRUE)`

References

See [events](#).

Examples

```
data(dice)
head(dice,10)
```

dichotomize

*Dichotomize***Description**

Converts factor or character columns of a data frame into a set of dichotomous (dummy/indicator) columns. Their names will correspond to the unique labels or texts of every category discovered across the selected variables. If multiple columns are provided, categories are unified per row.

Usage

```
dichotomize (data, variables, sep = "", regex = FALSE,
             text = c("none", "lower", "upper"),
             min = 1, length = 0, sort = TRUE,
             values = NULL, stopwords = NULL, nas = "None",
             sparse = FALSE, add = FALSE)
```

Arguments

data	A data frame containing the factor or textual columns to be converted. Cells can contain simple values or multiple events delimited by a separator.
variables	A character vector of column names to be converted into dichotomous vectors.
sep	A character string (or vector) used to divide columns with multiple events. If this separator is "" (default), every unique cell is treated as a single category. Use "C" to concatenate multiple columns before dichotomization.
regex	Logical. If FALSE (default), sep is matched exactly. If TRUE, sep is treated as a regular expression (e.g., "\\s+" to split by whitespace).
text	Character. Text transformation option. "none" (default) leaves text intact. "lower" or "upper" converts text to lower or upper case, respectively, and completely removes punctuation and numbers.
min	Integer or numeric. Converts to dichotomous vectors only the labels or texts that have a frequency greater than or equal to the value of this parameter. If the value is between 0 and 1, it is interpreted as a percentage of the total rows.
length	Integer. The strict maximum number of dichotomous columns generated, prioritizing the most frequent categories. 0 implies no limit.
sort	Logical. If TRUE, the new columns are ordered by their frequencies in descending order.
values	A character vector of specific labels or texts strictly selected for conversion into dichotomous columns.
stopwords	Character vector. An optional list of words to be removed from the text before dichotomization. Only applied if text is set to "lower" or "upper".
nas	Character. Variable name used to account for NA (missing) values across the set of variables. Set to NULL to ignore missing values.

sparse	Logical. If TRUE, returns a sparse matrix (requires the Matrix package) instead of a data.frame.
add	Logical. If TRUE, appends the newly generated dichotomous columns to the original input data.frame.

Value

A data frame composed of the original variables plus the added dichotomous columns, or a sparse matrix if sparse = TRUE.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca, and Luis Martinez Uribe, Fundacion Juan March.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# A character column
frame1 <- data.frame(A = c("Man", "Women", "Man", "Undet."))
dichotomize(frame1, "A", sep = "; ")

# A character column (with separator)
frame2 <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
dichotomize(frame2, "A", sep = "; ")

# A character column and another factor column (same separator)
frame3 <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"),
                    C = factor(c(1:4), labels = c("Paris", "New York",
                                                  "London; New York", "<NA>")))
dichotomize(frame3, c("A", "C"), sep = "; ")

# A set of simple character or factor (same levels) variables.
# In this case, you must use "C" separator.
frame4 <- data.frame(A = c("Man", "Women", "Man", "Undet.", NA),
                    B = c("Women", "Women", "Man", "Women", NA),
                    C = c(NA, NA, NA, "Man", NA))
dichotomize(frame4, c("A", "B", "C"), sep="C")

# Text mining example: Splitting by whitespace, removing punctuation/numbers,
# transforming to lowercase and filtering stopwords
frame5 <- data.frame(text_col = c("I love apples and 10 oranges!",
                                  "Apples are great.",
                                  "I hate bananas."))
dichotomize(frame5, "text_col", sep = "\\s+", regex = TRUE,
```

```
text = "lower", stopwords = c("i", "and", "are"))
```

distant

Distance matrix.

Description

Convert a similarity matrix into a distance matrix.

Usage

```
distant(s, t = FALSE)
```

Arguments

s	a similarity matrix
t	return the same matrix if t=FALSE

Details

For better results, use the parameter distance in [sim](#) function.

Value

A distance matrix.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# From a random incidence matrix I(25X4)
I <- matrix(rbinom(100, 1, .5), nrow = 25, ncol = 4,
  dimnames = list(NULL, c("A", "B", "C", "D")))
J <- sim(I, "Jaccard")
distant(J, t = TRUE)
#Same results
sim(I, "Jaccard", distance = TRUE)
```

dyncohort*Interactive graphs of dynamic cohorts.*

Description

dyncohort produces interactive graphs representing dynamic cohorts. Two periods or lives belongs to the same cohort if there are a difference of years in their start less or equal to a given number. In case of people's life, 15 or 25 are appropriate quantities to set. If year is equal to 0, a cohort is defined a those periods or lives that begin at the same year.

Usage

```
dyncohort( periods, name="name", start="start", fields=names(periods),  
           years=0, igraph=FALSE, ... )
```

Arguments

periods	a data frame with at least two vectors with name and start of the periods or lives.
name	name of the vector with names in the data frame.
start	name of the vector with starts in the data frame.
fields	vector of the names of the periods data frame to be taken into account.
years	number of years to be considered as length of the cohort.
igraph	produces an igraph object instead of a netCoin class.
...	Any netCoin argument.

Value

This function creates a timeCoin object (or igraph) and, if plotted, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

A netCoin object has three elements:

nodes	A data frame with the periods.
links	A data frame with the events.
options	A list of options for the interactive graph.

Note

Periods could be the life of people, in whose case start is their birth year.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[timeCoin](#) and [coexist](#)

Examples

```
# With sociologists data.
data(sociologists)
dyncohort(sociologists, start="birth")
# Consider 15 year dynamic cohort:
sociologists$picture <- system.file("extdata", sociologists$picture,
                                   package="netCoin") # copy path to the picture field
dyncohort(sociologists, start="birth", year=15, image="picture", imageName="name")
```

edgeList

Edge list.

Description

Convert a coincidence/similarity/distance matrix into an edge list form.

Usage

```
edgeList(data, procedures="Haberman",
         criteria="Z", level=.95, Bonferroni=FALSE,
         min=-Inf, max=Inf, support=-Inf, directed=FALSE,
         diagonal=FALSE, sort=NULL, decreasing=TRUE, pairwise=FALSE)
```

Arguments

data	a coin object, let's say an R matrix with frequencies and an attribute (n) giving the number of scenarios. In case of change of shape, data should be a matrix.
procedures	a vector of statistics of similarity. See below.
criteria	statistic to be use for selection criteria.
level	confidence level
Bonferroni	Bonferroni criterium of the signification test.
min	minimum value of the statistic to include the edge in the list.
max	maximum value of the statistic to include the edge in the list.
support	minimum value of the frequency of the coincidence to be edged
directed	includes same edges only once.
diagonal	includes auto-links
sort	sort the list according to the values of a statistic. See below
decreasing	order in a decreasing way.
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.

Details

Possible measures in procedures are

- Frequencies (f), Relative frequencies (x), Conditional frequencies (i), Coincidence degree (cc), Probable degree (cp),
- Expected (e), Confidence interval (con)
- Matching (m), Rogers & Tanimoto (t), Gower (g), Sneath (s), Anderberg (and),
- Jaccard (j), Dice (d), antiDice (a), Ochiai (o), Kulczynski (k),
- Hamann (ham), Yule (y), Pearson (p), odds ratio (od), Rusell (r),
- Haberman (h), Z value of Haberman (z),
- Hypergeometric p greater value (hyp).
- Convert a matrix into an edge list (shape).

Value

A data frame in which the two first columns are source and target. The rest of the columns are the different statistics explicited in funcs parameter.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# From a random incidence matrix I(25X4)
I<-matrix(rbinom(100,1,.5),nrow=25,ncol=4,
dimnames=list(NULL,c("A","B","C","D")))
C<-coin(I)
edgeList(C)
```

ess

Data: European Social Survey, Round-8.

Description

A sample size of 1,000 respondents from the European Social Survey, Round-8.

Usage

```
data("ess")
```

Format

A data frame with 1000 cases (respondents) and 5 variables:

Gender Gender (factor vector): Female, Male.

Age Age (recoded factor vector): 15-29, 30-30, 40-49, 50-59, 60-69, 70 and +.

Social participation Social participation (factor vector): No, Yes.

Political participation Political participation (factor vector): No, Yes.

cweight cweight (numeric vector): Cases weight.

References

ESS Round 8: European Social Survey Round 8 Data (2016). Data file edition 2.1. NSD - Norwegian Centre for Research Data, Norway - Data Archive and distributor of ESS data for ESS ERIC. [doi:10.21338/NSD-ESS8-2016](https://doi.org/10.21338/NSD-ESS8-2016).

Examples

```
data("ess")
head(ess,10)
```

events

Data: Attributes of the dice events.

Description

Data frame with the attributes of the events of dice.

Usage

```
data("events")
```

Format

A data frame with 10 observations on the following 4 variables:

name : a factor vector with 10 levels

label : a factor vector with 10 levels

frequency : a numeric vector

type : a factor vector with 2 levels

Source

```
data(dice); coin.dice<-coin(dice); asNodes(coin.dice)
```

References

See [dice](#).

Examples

```
data(events)
events
```

exhibit

Images in a grid gallery.

Description

exhibit produces an interactive image gallery (alternative display).

Usage

```
exhibit(nodes, name = NULL, label = NULL, subtitle = NULL, order = NULL,
  decreasing = FALSE, ntext = NULL,
  mainframeHeight = NULL, mainframeWidth = NULL, mainframeImage = 0,
  image = NULL, imageCopy = NULL, noFilterCols = NULL,
  zoom = NULL, scale = NULL, main = NULL, note = NULL,
  logo = TRUE, pages = c("bottom", "dynamic", "top"), mainposition = c("dynamic", "fixed"),
  frequencies = FALSE, tutorial = FALSE, tableButton = FALSE, export = FALSE,
  search = TRUE, colorScheme = 0, language = c("en", "es", "ca"), dir = NULL)
```

Arguments

nodes	a data frame with at least one column with the names of the gallery's elements.
name	name of the vector with names in the nodes data frame.
label	column name with image labels in the nodes data frame.
subtitle	column name with image subtitles in the nodes data frame.
order	name of the column with node order in the nodes data frame.
decreasing	decreasing or increasing sort of the nodes.
ntext	column name with html text in the nodes data frame.
mainframeHeight	a number between 0.1 and 1 to reduce mainframe height.
mainframeWidth	a number to proportionally change mainframe width.
mainframeImage	a number to display image in mainframe. 0 = auto (default), 1 = before text, 2 = top right
image	column name which indicates the image paths in the nodes data frame.
imageCopy	column name which indicates the image copyright and attribution in the nodes data frame.
noFilterCols	disable filtering by specified column names.
zoom	Deprecated. Replaced by argument 'scale'.
scale	a number between 0.1 and 1 as initial displaying scale.

main	upper title of the graph.
note	lower title of the graph.
logo	switch to FALSE if you want to hide netCoin logo.
pages	position of pagination info. In the header (top), footer (bottom), or depending on the scroll (dynamic).
mainposition	If 'fixed', the upper title will always be visible in the header, by default ('dynamic'), it will be hidden when scrolling.
frequencies	a logical value true if barplots representing node attributes frequencies will be added to the final graph.
tutorial	Should tutorial be displayed?
tableButton	A button will appear to show the data as a table.
export	A button will appear to export gallery as excel.
search	Should gallery search be displayed? (default: TRUE)
colorScheme	Select a color theme for display (0-11).
language	a character string indicating the language of the graph (en=english (default); es=spanish; ca=catalan).
dir	a character string representing the directory where the web files will be saved.

Value

Object of class gallery_rd3.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data("finches")
finches$species <- system.file("extdata", finches$species,
                             package="netCoin") # copy path to the species field
exhibit(finches, image="species")
```

expectedList	<i>Expected list.</i>
--------------	-----------------------

Description

Converts a coin object to a links data frame with coincidences and expected values.

Usage

```
expectedList(data, names = NULL, min = 1, confidence=FALSE)
```

Arguments

data is a coin object. See [coin](#)
names a character vector.
min minimum value of the statistic to include the edge in the list.
confidence add the confidence interval if TRUE.

Value

A links data frame with coincidences and expected values.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```

# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
expectedList(C) # edge data frame

```

families

Data: Italian families in the Renaissance.

Description

Data frame with the characteristics of powerful families of Renaissance Italy.

Usage

```
data("families")
```

Format

A data frame with 16 families (rows) and 6 characteristics.

name Family's name
f.Marriages number of marriage links
f.Business number of business links
wealth wealth's index
priorates number of priorates on control
seat At least priorate

Source

PADGETT, J. F. Y C. K. ANSELL (1993): "Robust Action and the Rise of the Medici, 1400-1434", in American Journal of Sociology, 98, 1259-1319. (<http://www.jstor.org/stable/2781822>)

Examples

```
data("families")
head(families)
```

finches

Data: Finches' attributes in Galapagos islands.

Description

Data frame with events as result.

Usage

```
data("finches")
```

Format

A data frame with 13 observations (pinches) and 4 variables (name and characteristics):

name : Genus and species of the finche

frequency : number of islands where the finche can be found

type : Genus of the finche

species : name of the file containing the picture of the finche

References

Sanderson, James (2000). Testing Ecological Patterns: A Well-known Algorithm from Computer Science Aids the Evaluation of Species Distributions. American Scientist, 88, pp. 332-339.

Examples

```
data(finches)
head(finches,10)
```

fromIgraph	<i>Produce interactive networks from igraph objects.</i>
------------	--

Description

fromIgraph produce an interactive network from an igraph object.

Usage

```
fromIgraph(G, ...)
```

Arguments

G	an igraph object.
...	Any netCoin argument.

Value

This function returns a netCoin object. If the 'dir' attribute is specified, the function creates a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
g <- igraph::make_ring(10)
fromIgraph(g)
```

Galapagos

Data: Finches' presence in Galapagos Islands.

Description

Data frame with absence(0) presence(1) of finches in the Galagos Islands.

Usage

```
data("Galapagos")
```

Format

A data frame with 17 localizations (islands) and 13 variables (Genus and species of the finches):

Geospiza magnirostris

Geospiza fortis

Geospiza fuliginosa

Geospiza difficilis

Geospiza scandens

Geospiza conirostris

Camarhynchus psittacula

Camarhynchus pauper

Camarhynchus parvulus

Platyspiza crassirostris

Cactospiza pallida

Cactospiza heliobates

Certhidea olivacea

References

Sanderson, James (2000). Testing Ecological Patterns: A Well-known Algorithm from Computer Science Aids the Evaluation of Species Distributions. American Scientist, 88, pp. 332-339.

Examples

```
data(Galapagos)
head(Galapagos, 10)
```

gallery	<i>Images in a grid gallery.</i>
---------	----------------------------------

Description

gallery produces an interactive image gallery.

Usage

```
gallery(nodes, name = NULL, label = NULL, color = NULL,
        border = NULL, ntext = NULL, info = NULL, infoFrame = c("right","left"),
        image = NULL, imageCopy = NULL, noFilterCols = NULL, zoom = NULL,
        scale = 1, itemsPerRow = NULL, main = NULL, note = NULL,
        showLegend = TRUE, frequencies = FALSE, labelTooltip = TRUE,
        cexTooltip = 1, help = NULL, helpOn = FALSE, tutorial = FALSE,
        description = NULL, descriptionWidth = NULL, roundedItems = FALSE,
        ntextctrl = FALSE, controls = 1:5, cex = 1, defaultColor = "#1f77b4",
        language = c("en", "es", "ca"), dir = NULL)
```

Arguments

nodes	a data frame with at least one column with the names of the gallery's elements.
name	name of the vector with names in the nodes data frame.
label	column name with image labels in the nodes data frame.
color	column name with image background color variable in the nodes data frame.
border	column name with image border width variable in the nodes data frame or a numeric vector.
ntext	column name with html text in the nodes data frame.
info	column name with information to display in a panel in the nodes data frame.
infoFrame	In which panel should the information be displayed? The left panel is only available if frequencies are not showing.
image	column name which indicates the image paths in the nodes data frame.
imageCopy	column name which indicates the image copyright and attribution in the nodes data frame.
noFilterCols	disable filtering by especified column names.
zoom	Deprecated. Replaced by argument 'scale'.
scale	a number between 0.1 and 10 as initial displaying scale.
itemsPerRow	number of items in each row.
main	upper title of the graph.
note	lower title of the graph.
frequencies	a logical value true if barplots representing node attributes frequencies will be added to the final graph.

labelTooltip	show label on mouseover.
cexTooltip	number indicating the amount by which plotting text should be scaled relative to the default in the label tooltip.
showLegend	a logical value true if the legend is to be shown.
help	a character string indicating a help text of the graph.
helpOn	Should the help be shown at the beginning?
tutorial	Should tutorial be displayed?
description	a character string indicating a description text for the graph.
descriptionWidth	a percentage indicating a width for the description panel (25 by default).
roundedItems	Display items with rounded borders.
ntextctrl	Previous tooltips will be closed unless you hold down the control key (FALSE by default).
controls	a numeric vector indicating which controls will be shown. 1 = topbar, 2 = xlsx exportation, 3 = table, 4 = netCoin logo, 5 = topbar buttons. NULL hide all controls, negative values deny each control and 0 deny all.
cex	number indicating the amount by which plotting text should be scaled relative to the default.
defaultColor	a character vector giving a valid html color for node representation.
language	a character string indicating the language of the graph (en=english (default); es=spanish; ca=catalan).
dir	a character string representing the directory where the web files will be saved.

Value

Object of class `gallery_rd3`.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data("finches")
finches$species <- system.file("extdata", finches$species,
  package="netCoin") # copy path to the species field
gallery(finches, image="species", main="Species in Galapagos Islands",
  note="Data source: Sanderson (2000)")
```

gallery3	<i>Images in a grid gallery.</i>
----------	----------------------------------

Description

gallery3 produces an interactive image gallery (alternative display).

Usage

```
gallery3(nodes, name = NULL, label = NULL, subtitle = NULL, order = NULL,
  decreasing = FALSE, ntext = NULL,
  mainframeHeight = NULL, mainframeWidth = NULL, mainframeImage = 0,
  image = NULL, imageCopy = NULL, imageRatio = NULL, noFilterCols = NULL,
  zoom = NULL, scale = NULL, main = NULL, note = NULL,
  tableButton = FALSE, export = FALSE,
  search = TRUE, cex = 1, language = c("en", "es", "ca"), dir = NULL)
```

Arguments

nodes	a data frame with at least one column with the names of the gallery's elements.
name	name of the vector with names in the nodes data frame.
label	column name with image labels in the nodes data frame.
subtitle	column name with image subtitles in the nodes data frame.
order	name of the column with node order in the nodes data frame.
decreasing	decreasing or increasing sort of the nodes.
ntext	column name with html text in the nodes data frame.
mainframeHeight	a number to proportionally change mainframe height.
mainframeWidth	a number to proportionally change mainframe width.
mainframeImage	a number to display image in mainframe. 0 = auto (default), 1 = before text, 2 = top right
image	column name which indicates the image paths in the nodes data frame.
imageCopy	column name which indicates the image copyright and attribution in the nodes data frame.
imageRatio	the aspect ratio of the images. Is the ratio of its width to its height. 3/4 by default.
noFilterCols	disable filtering by especified column names.
zoom	Deprecated. Replaced by argument 'scale'.
scale	a number between 0.1 and 1 as initial displaying scale.
main	upper title of the graph.
note	lower title of the graph.
search	Should gallery search be displayed? (default: TRUE)

tableButton	A button will appear to show the data as a table.
export	A button will appear to export gallery as excel.
cex	number indicating the amount by which plotting text should be scaled relative to the default.
language	a character string indicating the language of the graph (en=english (default); es=spanish; ca=catalan).
dir	a character string representing the directory where the web files will be saved.

Value

Object of class gallery_rd3.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
data("finches")
finches$species <- system.file("extdata", finches$species,
                               package="rD3plot") # copy path to the species field
gallery <- gallery3(finches, image="species")
## Not run:
plot(gallery)

## End(Not run)
```

get_panel_template	Create an html panel for nodes from different items (for galleries).
--------------------	--

Description

Create an html panel for nodes from different items (for galleries).

Usage

```
get_panel_template(data, title=NULL, description=NULL, img=NULL,
                  text=NULL, color="auto", cex=1, mode = 1)
```

Arguments

data	data frame which contains the data.
title	column name which contains the first title of the vignette.
description	column name which contains the main text of the vignette.
img	column name which contains the names of the image files.
text	column name which contains the main text of the vignette.

color	color of the panel (It also could be a column name which contains colors).
cex	number indicating the amount by which plotting text should be scaled relative to the default.
mode	There are two display modes.

Value

a character vector of html formatted panel.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
data("sociologists")
sociologists$picture <- system.file("extdata", sociologists$picture,
  package="netCoin")
sociologists$html <- get_panel_template(sociologists, title="name",
  description="birth_country",img="picture", text="school")
plot(gallery(sociologists, info="html"))

## End(Not run)
```

get_template	<i>Create a drop-down vignette for nodes from different items (for galleries).</i>
--------------	--

Description

Create a drop-down vignette for nodes from different items (for galleries).

Usage

```
get_template(
  data,
  title = NULL,
  title2 = NULL,
  text = NULL,
  img = NULL,
  wiki = NULL,
  width = 300,
  color = "auto",
  cex = 1,
  roundedImg = FALSE,
  mode = 1
)
```

Arguments

data	data frame which contains the data.
title	column name which contains the first title of the vignette.
title2	column name which contains the secondary title of the vignette.
text	column name which contains the main text of the vignette.
img	column name which contains the names of the image files.
wiki	column name which contains the wiki URL for the vignette.
width	length of the vignette's width.
color	color of the vignette's strip (It also could be a column name which contains colors).
cex	number indicating the amount by which plotting text should be scaled relative to the default.
roundedImg	Display images with rounded borders.
mode	2 display images next to the text. 1 by default.

Value

a character vector of html formatted vignettes.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
data("sociologists")
sociologists$picture <- system.file("extdata", sociologists$picture,
  package="netCoin")
sociologists$html <- get_template(sociologists, title="name",
  title2="birth_country", text="school",img="picture", mode=2)
plot(gallery(sociologists, ntext="html"))

## End(Not run)
```

get_template2	<i>Create a drop-down vignette for nodes from different items (for galleries).</i>
---------------	--

Description

Create a drop-down vignette for nodes from different items (for galleries).

Usage

```
get_template2(data, title=NULL, title2=NULL, text=NULL, wiki=NULL)
```

Arguments

data	data frame which contains the data.
title	column name which contains the first title of the vignette.
title2	column name which contains the secondary title of the vignette.
text	column name which contains the main text of the vignette.
wiki	column name which contains the wiki URL for the vignette.

Value

a character vector of html formatted vignettes.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
data("sociologists")
sociologists$html <- get_template2(sociologists, title="name",
  title2="birth_country", text="school")
plot(exhibit(sociologists, ntext="html"))

## End(Not run)
```

glmCoin

Regression Graphs

Description

produces a netCoin object from a set of glm regressions.

Usage

```
glmCoin(formulas, data, weights=NULL, pmax=.05,
  robust=FALSE, showArrows=TRUE,
  color="variable", lwidth="s.value", lcolor="estimate",
  language=c("en", "es", "ca"),
  igraph=FALSE, ...)
```

Arguments

<code>formulas</code>	A set of formulas separated, folowed by the family and a return. For example: <code>model <- "counts ~ outcome + treatment, poisson counts ~ outcome, poisson"</code> The possible families are <i>gaussian</i> (default), <i>poisson</i> , <i>binomial</i> and <i>multinomial</i> .
<code>data</code>	Data frame containing the variables in the model.
<code>weights</code>	Optional vector of weights to be used in the fitting process.
<code>pmax</code>	Selection of links with $\Pr(> z)$ less than p (one-tail by default).
<code>robust</code>	Robust estimation of standard errors if TRUE.
<code>showArrows</code>	a logical value true if the directional arrows are to be shown. Default = FALSE.
<code>color</code>	Nodes' attribute to be used for expressing color ("variable" by default).
<code>lwidth</code>	Nodes' attribute to be used for widht of arrows ("s.value" by default).
<code>lcolor</code>	Nodes' attribute to be used for color of arrows ("estimate" by default).
<code>language</code>	Language of the graph controls.
<code>igraph</code>	Produces an igraph object instead of a netCoin object if TRUE.
<code>...</code>	Any netCoin argument.

Value

This function creates a `netCoin` object (or `igraph`) and, if stated, a folder in the computer with an HTML document named `index.html` which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

The links of the graph carry the average marginal effects as GW contrasts (`estimate`, `std.error`, `p.value`: effects relative to the weighted overall mean) and also relative to the first category of each qualitative variable (`estimate.base`, `std.error.base`, `p.value.base`; for the base category itself the estimate is 0 and its standard error and p value are NA). Both sets can be displayed with [marginsTable](#).

The `netCoin` object also includes a data frame named `equations` with one row per declared equation and the following goodness-of-fit measures: `model` (formula), `family` (declared family), `n` (number of observations; the sum of weights in weighted models), `n_obs` (number of observed cases, regardless of weights), `r2` (deviance-based R2: $1 - \text{deviance} / \text{null deviance}$, which equals the classical R2 for gaussian models), `r2.mcfadden` (McFadden's pseudo-R2, NA for quasi families), `r2.nagelkerke` (Nagelkerke's pseudo-R2), `logLik`, `aic`, `bic` (NA for quasi families), `deviance`, `null.deviance` and `df.residual`. When `igraph=TRUE`, this data frame is attached as the graph attribute `equations`.

It also includes a list named `coeffs` with one data frame of regression coefficients per equation, in the same order as the rows of equations. Each data frame has one row per coefficient and response category, with columns `Target` (the response, or `response:category` in binomial and multinomial models), `term` (the name given by the fitted model), `Source` (the same term written as a node name, so that it can be matched against nodes and links), `factor` (whether the term involves a qualitative variable), `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high` and `sd.x` (the weighted standard deviation of the term, NA for the constant). The attributes `family`, `link`, `sd.y` (the weighted standard deviation of the response, NA unless it is on its own scale) and `latent.sd` (the standard deviation of the underlying latent variable, per `Target`, NA unless the link is logit or probit) record the scales needed to standardize or exponentiate the estimates. These are

the coefficients that `coefTable` displays; note that they are not the marginal effects carried by the links. When `igraph=TRUE`, this list is attached as the graph attribute `coeffs`.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
Dobson <- data.frame(counts=counts, outcome=outcome, treatment=treatment)
model <- "counts ~ outcome + treatment, poisson"
glmCoin(model,Dobson)

## Multinomial model: housing satisfaction in Copenhagen (Madsen, 1976)
data(housing, package="MASS")
g <- glmCoin("Sat ~ Infl + Type + Cont, multinomial", housing, weights="Freq")
g$equations
```

glmCoin2

Regression Graphs

Description

produces a `netCoin` object from a set of glm regressions.

Usage

```
glmCoin2(formulas, data, weights=NULL, pmax=.05,
          twotail=FALSE, showArrows=TRUE,
          frequency = FALSE, percentage = TRUE,
          color="variable", lwidth="z.value",
          circle= NA, language=c("en","es","ca"),
          igraph=FALSE, ...)
```

Arguments

<code>formulas</code>	A set of formulas separated, folowed by the family and a return. For example: <code>model <- "counts ~ outcome + treatment, poisson counts ~ outcome, poisson"</code>
<code>data</code>	Data frame containing the variables in the model.
<code>weights</code>	Optional vector of weights to be used in the fitting process.
<code>pmax</code>	Selection of links with $\Pr(> z)$ less than p (one-tail by default).
<code>twotail</code>	Logical value indicating if twotail test must be appied. Default=FALSE.
<code>showArrows</code>	a logical value true if the directional arrows are to be shown. Default = FALSE.

frequency	a logical value true if frequencies are to be shown. Default=FALSE.
percentage	a logical value true if percentages are to be shown. Default=TRUE.
color	Nodes' attribute to be used for expressing color ("variable" by default).
lwidth	Nodes' attribute to be used for width of arrows ("z.value" by default).
circle	Degree of rotation in case of fixed circled dependent variables.
language	Language of the graph controls.
igraph	Produces an igraph object instead of a netCoin object if TRUE.
...	Any netCoin argument.

Value

This function creates a netCoin object (or igraph) and, if stated, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
Dobson <- data.frame(counts=counts, outcome=outcome, treatment=treatment)
model <- "counts ~ outcome + treatment, poisson"
glmCoin2(model,Dobson)
```

incTime	<i>Time incidences.</i>
---------	-------------------------

Description

Convert a data frame with two numbers (normally a beginning year and end year) into an incidences matrix whose rows are the intermediate numbers, and whose columns are the content of the names column.

Usage

```
incTime(data, name = "name", beginning = "birth", end= "death")
```

Arguments

data	a data frame a name and two numbers.
name	Column with the names (default= "name").
beginning	Column with the beginning number to include (default= "birth").
end	Column with the end number to include (default= "death").

Value

A data frame in which the two first columns are source and target. The rest of the columns are $\text{sim.} = (1 + \text{threshold} - \text{real difference})$ and $\text{dist.} = (\text{difference between numbers})$

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# From sociologists data
data("sociologists")
head(incTime(sociologists))[,1:5]
```

layoutCircle	<i>Produce a circle layout of any number of nodes.</i>
--------------	--

Description

layoutCircle produces a circle layout of any number of nodes.

Usage

```
layoutCircle(N, nodes=seq_len(nrow(N)), deg=0, name=NULL)
```

Arguments

N	a data frame of nodes.
nodes	a vector specifying nodes.
deg	degrees to rotate.
name	name of column with node names.

Value

This function returns the input data frame of nodes with the resulting layout applied.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
A <- data.frame(name=letters)
L <- layoutCircle(A, name="name")
netCoin(A, layout=L)
```

layoutDist	<i>Distance between two planes of a scattergram.</i>
------------	--

Description

layoutDist measures how far one and the same node travels from one plane of coordinates to another, added up over the nodes and divided by how many they are.

Usage

```
layoutDist(a, b, align=c("cloud", "procrustes", "canvas", "none"), weights=NULL,
           normalize=TRUE, who="layoutDist")
```

Arguments

a, b	two planes, each a matrix or data.frame of coordinates holding one row per node, in the same order.
align	how the two planes are put on an equal footing before measuring, and with it what is being asked. See the details.
weights	a vector holding what every node counts for, applied to the sum and to the centring, the scaling and the rotation alike. NULL (default) counts every node once; where the nodes are patterns it is worth passing the cases each one stands for, so that a rare pattern does not move as much as a common one.
normalize	whether to divide the sum by the number of nodes compared, or by the total of the weights where there are any. TRUE by default, which makes the figure how far any one node travels: it no longer depends on the size of the survey, so two different objects are comparable, and under align="canvas" it falls between 0 and sqrt(2), the diagonal of the canvas. FALSE returns the plain sum.
who	the name errors are issued under.

Details

Every plane comes in its own units: the coordinates of a tsne reduction reach 60 where those of a classical one stop at 3, and nmf or a pair of observed variables live besides in the positive quadrant. Adding up the distances as they come measures little else than that difference of units, so the two planes are put on an equal footing first. How they are levelled decides what is being asked:

"cloud" centres every plane on its centroid and divides it by its root mean square radius. This strips translation and size away; what is left is a difference of configuration, orientation included. The divisor is one for the whole plane rather than one per column: standardising column by column would stretch the second axis until it matched the first, which undoes what a reduction states, that its first direction carries more variation than its second.

"procrustes" does the above and rotates besides the first plane, reflecting it where that helps, until it lies as close to the second as it can. It measures shape alone: two reductions ordering the cases alike but coming out rotated or mirrored end up at nearly no distance, which is the reasonable reading wherever the orientation of the axes means nothing, as in tsne or umap. No scaling factor

of the fit is added, though the classical Procrustes statistic carries one: the sizes have been levelled already, and that factor would break the symmetry a matrix of distances between planes needs.

"canvas" brings every column to [0, 1] by its minimum and its maximum, which is what rD3plot does when it sets the domain of each axis to the range of the coordinates and its own range to the width or the height of the canvas. It measures the picture the reader is shown rather than the cloud behind it, and two planes differing only in the scale of their axes come out at zero. The canvas is taken square, as the shape of the window is not known from R.

"none" leaves the coordinates as they are, which fits where the units are already comparable, as between two pairs of variables measured from 0 to 100.

Rows holding NA in either plane are dropped, and the alignment is computed on the rows about to be compared rather than on the whole of them.

Value

A single number, the distance between the two planes. It carries as attributes the rows compared ("rows") and those dropped ("dropped"), the alignment applied ("align"), the divisor applied to each plane ("scales", one per plane under "cloud" and "procrustes", one per axis under "canvas") and, with align="procrustes" and planes of two columns, the rotation in degrees ("angle") and whether there was a reflection ("reflection").

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[layoutsDist](#), [addAxes](#), [surScat](#)

Examples

```
## Not run:
data(iris)
S <- surScat(iris, variables=names(iris)[1:4], type="pca", nclusters=3)
P <- prcomp(iris[,1:4], scale.=TRUE)$x
S <- addAxes(S, P[,3:4])
L <- S$layouts

layoutDist(L[[1]], L[[2]])           # the cloud
layoutDist(L[[1]], L[[2]], align="procrustes") # shape alone
layoutDist(L[[1]], L[[2]], align="canvas")    # the picture

z <- layoutDist(L[[1]], L[[2]], align="procrustes")
attr(z, "angle")
attr(z, "reflection")

## End(Not run)
```

layoutGrid	<i>Produce a layout of any number of nodes.</i>
------------	---

Description

layoutGrid produces a grid layout of any number of nodes.

Usage

```
layoutGrid(N,string,name=NULL,byrow=FALSE)
```

Arguments

N	a data frame of nodes.
string	a character vector specifying grouped nodes.
name	name of column with node names.
byrow	logical. If 'FALSE' (the default) the layout is filled by columns, otherwise the layout is filled by rows.

Value

This function returns the input data frame of nodes with the resulting layout applied.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
A <- data.frame(name=letters)

L <- layoutGrid(A,"a,b,c,d,e.f,g,h,i,j.k,l,m,n,o,p.q,r,s,t,u.v,w,x,y,z","name")

netCoin(A,layout=L)
```

layoutsAlign	<i>Planes of a scattergram laid over one another</i>
--------------	--

Description

layoutsAlign brings the planes of a netCoin object from surScat to a common footing, as [layoutsDist](#) does to measure how far apart they are, and hands back the transformed coordinates themselves.

Usage

```
layoutsAlign(x, align=c("procrustes", "cloud", "canvas", "none"), to=1, gpa=FALSE,
             weights=NULL, which=NULL, suffix=NULL, tol=1e-10, maxit=100,
             who="layoutsAlign")
```

Arguments

<code>x</code>	a 'netCoin' object from <code>surScat</code> , or a list of planes, each a matrix or data.frame of two columns of coordinates holding one row per node.
<code>align</code>	how the planes are put on an equal footing: "procrustes" (default) centres them, brings them to the same size and rotates (or reflects) each one onto the reference; "cloud" does the same without rotating; "canvas" stretches every axis to the same range, as the viewer does; "none" leaves them as they are. See layoutDist .
<code>to</code>	the reference plane, as its name or its index among all the planes of <code>x</code> . The first one by default, which in a scattergram from <code>surScat</code> is the plane it was drawn with (for instance "PCA" or "MCA"). The names are those printed under "Planes" by <code>print(x)</code> , or <code>names(x\$layouts)</code> : a reduction taken from a 'looking4clusters' object by <code>addAxes</code> keeps its own name, such as "pca" or "tsne", unless name was given there.
<code>gpa</code>	with <code>align="procrustes"</code> , rotate every plane towards the mean of them all (generalized Procrustes analysis) instead of towards <code>to</code> . The reference then only sets the orientation and the units in which the whole set is handed back.
<code>weights</code>	what every node counts for in the fit, as in layoutsDist .
<code>which</code>	the planes to align, as names or as indices. NULL (default) takes every one of them. The reference need not be among them; planes left out are kept as they were.
<code>suffix</code>	only for a 'netCoin' object. NULL (default) replaces its planes by the aligned ones. A string keeps the original planes and places each aligned one right after its original, so that the selector of the viewer offers them side by side, named as the original followed by <code>suffix</code> (with <code>suffix=""</code> , "sepal" gives "sepal"). The titles of its axes take the suffix too ("PC1" gives "PC1'"), as a rotated axis is no longer the original one. The reference plane is not added again, as it is left unchanged, except with <code>gpa=TRUE</code> . Whether or not a suffix is given, an axis titled with the percentage of variance of its factor, as in "PC1 (73.0%)", is titled with the percentage the rotated axis holds instead ("PC1' (71.4%)"): the factors being uncorrelated, it is worked out from theirs and the rotation, and the two axes of the plane keep the same total.
<code>tol, maxit</code>	convergence tolerance and maximum number of iterations of the generalized Procrustes fit.
<code>who</code>	the name errors and messages are issued under.

Details

The fit is estimated on the rows that are complete in every plane involved, the same ones for all of them, and then applied to every row of each plane: a node missing from one plane keeps its place in the others.

The result is expressed in the units of the reference plane, which is therefore left just as it was (except under `gpa=TRUE`, where every plane moves towards the consensus). A plane compared with it under `align="procrustes"` is thus turned to be read on the same axes, and the distance `layoutsDist` reports between them is the mean distance between the two sets of coordinates divided by the scale of the reference.

Under `gpa=TRUE` no plane is favoured over the others. The mean has no orientation of its own, so the whole set is finally turned, as one, to lie as close as possible to the reference.

Value

When `x` is a 'netCoin' object, the same object with the aligned planes in place of the original ones in its `$layouts` (or each one next to its original, when `suffix` is given), the first plane also as its `fx/fy` columns, and the cluster centroids in `$clusters` computed again on the new coordinates. The details of the alignment go in its attribute "alignment".

When `x` is a list of planes, a named list holding the aligned ones, with those details as attributes of the list.

The details are: the method (`align`), the reference plane (`to`), whether `gpa` was used, the rows the fit was estimated on (`rows`) and those left out (dropped), the center, scale and rotation matrix applied to every plane, and, with `align="procrustes"` and two columns, the angle of each rotation in degrees and whether it involved a reflection. With `gpa=TRUE` there are also the consensus plane (the mean of all the aligned ones), the number of iterations and whether the fit converged.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

`layoutsDist`, `layoutDist`, `addAxes`, `surScat`

Examples

```
## Not run:
data(iris)
S <- surScat(iris, variables=names(iris)[1:4], type="pca", nclusters=3)
P <- prcomp(iris[,1:4], scale.=TRUE)$x
S <- addAxes(S, P[,3:4])
S <- addAxes(S, as.matrix(iris[,1:2]), name="sepal")

# every plane turned onto PCA, the one drawn first
A <- layoutsAlign(S)
A
attr(A, "alignment")$angle

# onto another plane, by its name
A <- layoutsAlign(S, to="sepal")

# the original planes kept, the aligned ones added as "PC3-PC4" and "sepal"
A <- layoutsAlign(S, suffix="")
names(A$layouts)
```

```
# towards the mean of them all
G <- layoutsAlign(S, gpa=TRUE)
head(attr(G, "alignment")$consensus)

# the coordinates alone, from a list of planes
L <- layoutsAlign(S$layouts, which=c("sepal", "PC3-PC4"), to=1)

## End(Not run)
```

layoutsDist

Distances between all the planes of a scattergram.

Description

layoutsDist measures how far apart every pair of planes of a netCoin object from surScat lies, and hands the result back as the dist object hclust, cmdscale and the like expect.

Usage

```
layoutsDist(x, align=c("cloud", "procrustes", "canvas", "none"), weights=NULL,
            which=NULL, normalize=TRUE, nperm=0,
            value=c("distance", "relative", "p", "chance"), digits=3, who="layoutsDist")
```

Arguments

x	a 'netCoin' object from surScat, or a list of planes, each a matrix or data.frame of two columns of coordinates holding one row per node.
align	how the two planes are put on an equal footing before measuring, and with it what is being asked. See layoutDist .
weights	what every node counts for. The name of a numeric node column, or a vector holding one value per node or per case. NULL (default) weighs every node by the cases it stands for when the nodes are patterns, and counts one per node otherwise. weights=1 forces the plain count of nodes.
which	the planes to compare, as names or as indices. NULL (default) takes every one of them.
normalize	whether to divide by the number of nodes compared, or by the total of the weights where there are any, so that every entry is how far any one node travels. TRUE by default.
nperm	the number of permutations used to find what distance chance alone yields. 0 (default) computes none. See Details.
value	what the dist holds: "distance" (default), the distance observed; "chance", the mean distance over the permutations; "relative", the distance divided by that mean, 0 for the same plane and about 1 for planes no more alike than by chance; "p", the probability of a distance as small as the one observed if the two planes had nothing to do with each other. All but the first need nperm.

digits	the number of decimals shown when the result is printed, 3 by default, always in fixed notation. The figures themselves are not rounded. <code>print(x, digits=)</code> overrides it for a single printing.
who	the name errors and messages are issued under.

Details

A scattergram may hold several planes: the one it is drawn with, the reductions of a 'looking4clusters' object and any pair of variables added through [addAxes](#). All of them hold one row per node, in the same order, so node i is the same case in every one of them and it makes sense to ask how far it travels from one to another.

The distances are computed pair by pair with [layoutDist](#), which is where the alignments and the weighting are described. Being a plain dist, the planes can be clustered with `hclust` or placed against one another with `cmdscale` with no further ado.

Two readings are worth keeping apart. Under `align="cloud"` a zero means two planes hold the same configuration of points; under `align="canvas"` it means the viewer shows them as the same picture, which is a weaker statement, as `rD3plot` stretches the range of each axis to the width or the height of the canvas and so draws as one two planes differing in the scale of their axes. A distance collapsing under `align="procrustes"` points to two reductions agreeing but for their orientation.

The rotations are handed back as a full square matrix rather than inside the dist because they are not symmetric: rotating A onto B is undoing the rotation of B onto A, so the angle changes sign. A reflection, on the contrary, is its own inverse, and the angle of a reflected pair is the same both ways. Whether a pair is reflected comes as a symmetric square matrix alongside.

How large a distance is depends on the alignment. Under "cloud" and "procrustes" it is measured in radii of the cloud, and two unrelated planes lie about 1.25 apart; somewhat less under "procrustes" with few nodes, since a rotation always finds some likeness by chance. Under "canvas" the level of chance falls as the nodes grow in number. `nperm` settles that level for the planes at hand: for each pair, the rows of one plane are shuffled `nperm` times, which keeps the shape of both clouds but breaks the link between a node in one and the same node in the other, and the distance is measured again every time. The p-value is $(1 + m)/(nperm + 1)$, where m is the number of shuffles coming as close as the observed planes or closer, so it can go no lower than $1/(nperm + 1)$. A small p says the planes are related, not that they are alike: with many nodes a weak resemblance is enough, and the relative distance tells how strong it is.

To obtain the aligned coordinates rather than the distances between them, see [layoutsAlign](#).

Value

An object of class 'layoutsDist', which is also a 'dist', holding one distance per pair of planes, with the alignment recorded as its "method". It carries as attributes the rows compared per pair ("rows"), the weights actually applied ("weights") and where they came from ("wSource") and, with `align="procrustes"`, the rotation in degrees ("angle") and whether there was a reflection ("reflection"), both as square matrices.

With `nperm` above zero, the dist holds whatever value asks for, and the other three (among "distance", "relative", "p" and "chance") go along as attributes, each a dist of the same shape. The attribute "value" says which one the object itself holds.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[layoutDist](#), [layoutsAlign](#), [addAxes](#), [surScat](#)

Examples

```
## Not run:
data(iris)
S <- surScat(iris, variables=names(iris)[1:4], type="pca", nclusters=3)
P <- prcomp(iris[,1:4], scale.=TRUE)$x
S <- addAxes(S, P[,3:4])
S <- addAxes(S, as.matrix(iris[,1:2]), name="sepal")

D <- layoutsDist(S)
D
as.matrix(D)
plot(hclust(D))

# what the reader is actually shown
layoutsDist(S, align="canvas")

# only some of the planes, and the rotations behind the shape
layoutsDist(S, which=c("sepal", "PC3-PC4"))
Q <- layoutsDist(S, align="procrustes")
round(attr(Q, "angle"))
attr(Q, "reflection")

# against chance: distance relative to it, and probability
set.seed(1)
R <- layoutsDist(S, align="procrustes", nperm=999, value="relative")
R
attr(R, "p")

## End(Not run)
```

links

Data: Links between Italian families in the Renaissance.

Description

Data frame with the marriage and business links.

Usage

```
data("links")
```

Format

A data frame with 36 links (rows) amongst 16 Italian families in the Renaissance.

Albizzi

Acciaiuoli

Barbadori

Bischeri

Castellani

Guadagni

Lamberteschi

Medici

Pazzi

Peruzzi

Ridolfi

Salviati

Strozzi

Tornabuoni

Ginori

Pucci

link Type of link: marriage or business

Source

PADGETT, J. F. Y C. K. ANSELL (1993): "Robust Action and the Rise of the Medici, 1400-1434", in American Journal of Sociology, 98, 1259-1319. (<http://www.jstor.org/stable/2781822>)

Examples

```
data("links")
head(links)
```

logCoin

Networked log-linear models.

Description

logCoin produces a network object from loglinear models parameters.

Usage

```
logCoin(data, variables=names(data), exogenous=NULL, noFirstCat= NULL,
weight=NULL, order= 2, pairwise=FALSE, twotails = FALSE,
pmax = 0.05, frequency = FALSE, percentage = FALSE,
directed=FALSE, igraph=FALSE, ...)
```

Arguments

data	a data frame.
variables	a vector of names of variables included in the previous data frame.
exogenous	a vector of names of variables whose relations amongst them are of no interest. None by default.
noFirstCat	a vector of names of dichotomous variables to appear without the category (no characteristic).
weight	a vector of weights. Optimal for dataframed tables.
order	maximum order parameters to be included in the loglinear model (default=2)
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.
twotails	Application of twotail tests to the parameters (default: FALSE, i.e., onetail)
pmax	maximum value of the statistic to include the edge in the list. By default is 0.05, but 0.5 is recommended if data has not been sampled.
frequency	a logical value true if frequencies are to be shown. Default=FALSE.
percentage	a logical value true if percentages are to be shown. Default=TRUE.
directed	includes arrows to the links (target would be the categories of first mention variables).
igraph	Produces an igraph object instead of a netCoin object if TRUE.
...	Any netCoin argument.

Value

This function creates a netCoin object (or igraph) and, if stated, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# A n=1,000 sample from the European Social Survey.(Round 8, 2016)
data("ess")
logCoin(ess, c("Social participation", "Political participation"),
  noFirstCat=c("Social participation", "Political participation"),
  exogenous=c("Age", "Gender"), weight = "cweight",
  order=3, pmax=.05,percentage = TRUE, frequency = TRUE ) # network object
```

looking4clusters	<i>Clustering determination and visualization</i>
------------------	---

Description

looking4clusters Creates interactive clustering visualization plots from a SingleCellExperiment object, Seurat object or a matrix. It can also apply dimension reduction and clustering techniques.

Usage

```
looking4clusters(data, groups = NULL,
  components = FALSE, running_all = TRUE, distance = "euclidean",
  agglomeration = "complete", selectedk = NULL, perplex = 30,
  maxIter = 1000, threads = NULL, force_execution = FALSE)
```

Arguments

data	A numeric data.frame or matrix where variables are in columns and samples are in rows.
groups	Factor/Vector which defines an input category for each sample which will be visualized in the output plot.
components	If TRUE, clustering algorithms are applied to main components obtained with PCA.
running_all	If TRUE, applies dimension reduction and clustering techniques if the input is a matrix like object.
distance	the distance measure to be used with hierarchical clustering algorithm. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.
agglomeration	the agglomeration method to be used with hierarchical clustering algorithm. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
selectedk	Number of expected clusters given by the user. By default, the algorithm will run clustering algorithms with a number of clusters between 2 and 10. If selectedk is specified, a range of 5 units around the 'selectedk' will be used as number of clusters.
perplex	The perplexity parameter used for t-SNE algorithm execution (should not be bigger than $3 * \text{perplexity} < \text{nrow}(X)-1$). This value effectively controls how many nearest neighbors are taken into account when constructing the embedding in the low-dimensional space (default: 30)
maxIter	The number of iterations used for t-SNE algorithm execution (default: 1000)
threads	The number of CPU threads for calculating a distance matrix. Default value is the amount of CPU cores available on the system.
force_execution	force the execution of some methods that could cause performance problems with large matrices.

Value

Object of class looking4clusters.

Author(s)

David Barrios, Angela Villaverde and Carlos Prieto. Bioinformatics Service of Nucleus, University of Salamanca.

Examples

```
obj <- looking4clusters(iris[,1:4], groups=iris[,5], threads = 2)
```

lower

Similarity/distance matrix display.

Description

Display the lower part of a matrix with a specified number of decimals.

Usage

```
lower(matrix, decimals = 3)
```

Arguments

matrix	a symmetric similarity/distance matrix
decimals	number of decimals to be displayed

Value

A data frame of characters.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# From a random incidence matrix I(25X4)
I <- matrix(rbinom(100, 1, .5), nrow = 25, ncol = 4,
  dimnames = list(NULL, c("A", "B", "C", "D")))
lower(sim(I, "Jaccard"), 2)
```

makeT	<i>Create a vignette (for galleries).</i>
-------	---

Description

Create a vignette from w_EntityInfo from wikiTools package (for galleries).

Usage

```
makeT(data, imageDir=NULL, imageDefault=NULL, language=c("en","es","ca"))
```

Arguments

data	A data-frame created with w_EntityInfo from wikiTools package.
imageDir	A directory where images will be found. The name must be the 'entity' column content follow by the image extension and the allowed formats are jpg, jpeg, gif, png and svg.
imageDefault	This image will be assigned if 'entity' image is not found in imageDir or no imageDir is specified.
language	a character string indicating the language of the graph (en=english (default); es=spanish; ca=catalan).

Value

a data frame wich includes a column of html formatted vignettes called 'info'.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

makeT2	<i>Create a vignette (for galleries).</i>
--------	---

Description

Create a vignette from some specific information supplied (for galleries).

Usage

```
makeT2(entityLabel, image=NA, entityDescription=NA,
       byear=NA, bplace=NA, bcountry=NA, dyear=NA, dplace=NA, dcountry=NA,
       gender=NA, occupation=NA, language=c("en","es","ca"))
```

Arguments

entityLabel	A character string of names.
image	A character string of image paths.
entityDescription	A character string of descriptions.
byear	A character string of birth years.
bplace	A character string of birth places.
bcountry	A character string of birth countries.
dyear	A character string of death years.
dplace	A character string of death places.
dcountry	A character string of death countries.
gender	A character string of genders.
occupation	A character string of occupations.
language	a character string indicating the language of the graph (en=english (default); es=spanish; ca=catalan).

Value

a data frame wich includes a column of html formatted vignettes called 'info'.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

marginsTable	<i>Marginal Effects Table from Regression Graphs</i>
--------------	--

Description

Builds a marginal-effects table from the object returned by [glmCoin](#), using only the information stored in it: nodes (frequency, percentage and mean of each category), links (average marginal effects, standard errors and p values) and the equations data frame (n and pseudo-R2 of each model). Supports several dependent variables (including multinomial responses), each producing its own block of columns, and can put a transformed dependent back on its original scale.

Usage

```
marginsTable(G, by = c("category", "variable"),
  criterion = c("p", "effect"), contrast = c("gw", "base"),
  columns = c("n", "marg", "effect", "se", "sig"),
  transform = NULL, level = 0.95,
  sig = c(0.05, 0.01, 0.001), note = FALSE,
  digits = 1, footer = c("n", "r2"), file = NULL, format = NULL)
```

Arguments

G	A netCoin object created by glmCoin .
by	"category" produces one row per category of the explanatory variables; "variable" collapses to one row per variable, using its representative category.
criterion	How the representative category is chosen when by="variable": "p" (smallest p value) or "effect" (largest absolute effect).
contrast	"gw" (default) shows marginal effects relative to the weighted overall mean (GW contrasts); "base" shows them relative to the first (base) category of each qualitative variable, whose row displays ref. (slopes of numeric variables are the same under both). Requires the estimate.base columns stored in G\$links by the current version of glmCoin .
columns	Which columns to display: "n" is the global frequency column, and "marg" (marginal level), "effect" (marginal effect), "se" (standard error), "ci" (confidence interval of the effect) and "sig" (significance stars) form the block repeated for each dependent variable. At least one block column is required. The default is c("n", "marg", "effect", "se", "sig"), which becomes c("n", "marg", "effect", "ci", "sig") when transform is given and columns is left untouched.
transform	Function, or the name of one, putting the dependent back on its original scale; NULL (default) leaves the table on the scale of the model. Shortcuts: "expm1" (response modelled as $\log_1 p(y)$), "exp" (as $\log(y)$), "proportion" (percentages to proportions), "percent" (the reverse), "inverse" ($1/y$) and "expit" (log-odds to probability); any other name is looked up with match.fun , so "sqrt" or a user function also work. The marginal level becomes $g(\text{marg})$ and the effect the difference between two transformed levels, $g(\text{marg}) - g(\text{marg} - \text{effect})$, which is exact for any monotone g and reduces to $a \cdot \text{effect}$ when g is linear. Note that a standard error cannot be carried through a non-linear function: the interval of the "ci" column is obtained by transforming its ends, and is therefore asymmetric, whereas the "se" column, if requested, holds the first-order (delta-method) approximation.
level	Confidence level of the "ci" column, 0.95 by default; it also names the column (CI95%, CI99%, ...).
sig	Numeric vector with the significance thresholds of the stars, sorted internally in decreasing order: one star for p below the largest threshold, two for the next, and so on (any number of thresholds is allowed, e.g. <code>sig=c(.1, .05, .01, .001)</code> for a four-star scale).
note	If TRUE, a legend with the meaning of the stars (built from sig) is added below the table, both when printing and in the exported formats.
digits	Number of decimals for percentages and effects.
footer	Character vector with the columns of G\$equations to show as bottom rows, one value per model (multinomial responses share a single value across their category blocks). Available columns: n, n_obs, r2, r2.mcfadden, r2.nagelkerke, logLik, aic, bic, deviance, null.deviance, df.residual and family. Use NULL to omit the footer.
file	Optional file name to export the table. The format is inferred from the extension (.txt, .tex or .csv) unless format is given.

format Export format: "text", "latex" or "csv" (the latter opens directly in Excel). If file is missing, a default name is used.

Value

A character matrix of class `marginsTable`, printed with grouped headers (one block of columns per dependent variable: marginal level, marginal effect, standard error or confidence interval, and significance stars). The bottom rows show the statistics selected in footer (by default the `n` and the pseudo-R2 of each equation), taken from `GSequations`. When exporting, the table is returned invisibly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

See Also

[glmCoin](#)

Examples

```
data(birthwt, package="MASS")
birthwt$smoke <- factor(birthwt$smoke, labels=c("no","yes"))
birthwt$race <- factor(birthwt$race, labels=c("white","black","other"))
g <- glmCoin("low ~ smoke + race, binomial", birthwt)
marginsTable(g)
marginsTable(g, by="variable")

## the same table with proportions instead of percentages: a linear
## transformation, so effects and standard errors are simply rescaled
marginsTable(g, transform="proportion", digits=3)

## a dependent modelled in logarithms, back on its original scale:
## levels and effects in grams, with asymmetric confidence intervals
birthwt$lbwt <- log1p(birthwt$bw)
glog <- glmCoin("lbwt ~ smoke + race", birthwt)
marginsTable(glog)
marginsTable(glog, transform="expm1", digits=0)

## Poisson model: Dobson (1990) Page 93, Randomized Controlled Trial
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
Dobson <- data.frame(counts=counts, outcome=outcome, treatment=treatment)
gp <- glmCoin("counts ~ outcome + treatment, poisson", Dobson)
marginsTable(gp, contrast="base", note=TRUE)
```

mobileEdges

Mobile Edges.

Description

Convert a data frame with one number (normally a year) into an edge list form with those whose numbers (years) have a difference lower or equal to a quantity.

Usage

```
mobileEdges(data, name = 1, number = 2, difference=0)
```

Arguments

data	a data frame with a name and a number (year).
name	Column with the names (default= first column).
number	Column with the number (year) to compare (default= second column).
difference	Minimum difference between numbers of every two pair of names to create the edge or link (default=15).

Value

A data frame in which the two first columns are source and target. The rest of the columns are $sim. = (1 + threshold - real\ difference)$ and $dist. = (difference\ between\ numbers)$

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# From a random incidence matrix I(25X4)
data("sociologists")
mobileEdges(sociologists)
```

multigraphCreate

Produce interactive multi graphs.

Description

multigraphCreate produce an interactive multi graph.

Usage

```
multigraphCreate(..., mode = c("default", "parallel", "frame"),
  mfrow = c(1,2),
  frame = 0, speed = 50, loop = FALSE, lineplots = NULL,
  dir = NULL, show = FALSE)
```

Arguments

...	rD3plot graphs (network_rd3, barplot_rd3, timeplot_rd3) objects or html "directories".
mode	a string specifying the displaying mode. The "default" displays graphs one by one, "parallel" splits screen and "frame" allows dinamic graphs in time.
mfrow	a vector of the form 'c(nr, nc)'. Subsequent graphs will be drawn in an 'nr'-by-'nc' array on the device by rows. (Only applied in 'parallel' mode)
frame	number of frame to start a dynamic network.
speed	a percentage for frame speed in dynamic networks.
loop	allowing continuous repetition.
lineplots	a character vector giving the node attributes to show as lineplots.
dir	a "character" string representing the directory where the graph will be saved.
show	a logical value true if the graph is to be shown. Default = FALSE.

Value

This function returns a `mGraph` object. The function creates a folder in your computer with an HTML document named `index.html` which contains the graph. This file can be directly opened with your browser.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
  "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C, c("frequency", "expected", "haberman")) # edge data frame
bC <- barCoin(data, dichotomies = "_all") # barCoin object
cC <- barCoin(data, dichotomies = "_all", expected = TRUE) # barCoin object
nC <- netCoin(N, E) # netCoin object
multi <- multigraphCreate("Bar graph" = bC,
  "Conditional bar graph" = cC,
  "Net graph" = nC)

## Not run:
plot(multi)
```

```
## End(Not run)
```

multiPages	<i>Produces a gallery of 'netCoin' graphs.</i>
------------	--

Description

multiPages produces a gallery page to explore multiple 'netCoin' graphs.

Usage

```
multiPages(x, title = NULL, columns = NULL, imageSize = NULL,
  description = NULL, note = NULL,
  cex = 1, dir = tempDir(), show = FALSE)
```

Arguments

x	is a mGraph object. See multigraphCreate
title	the text for a main title.
columns	a numeric vector giving the number of columns to display items in gallery. Default = 3.
imageSize	a numeric vector giving the size of images in gallery. Default = 75.
description	a description text for the gallery.
note	a footer text for the gallery.
cex	number indicating the amount by which plotting text should be scaled relative to the default. Default = 1.
dir	a "character" string representing the directory where the graph will be saved.
show	a logical value true if the graph is to be shown. Default = FALSE.

Value

The function creates a folder in your computer with an HTML document named index.html which contains the graph. This file can be directly opened with your browser.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))

data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C,c("frequency","expected","haberman")) # edge data frame
bC<- barCoin(data,dichotomies="_all") # barCoin object
cC<- barCoin(data,dichotomies="_all",expected=TRUE) # barCoin object
nC<- netCoin(N,E) # netCoin object
multi <- multigraphCreate("Bar graph" = bC,
                          "Conditional bar graph" = cC,
                          "Net graph"=nC)

## Not run:
multiPages(multi,"Some graphs",show=TRUE)

## End(Not run)
```

netCoin

Networked coincidences.

Description

netCoin produces a netCoin object of coincidences. Its input has to be two data.frames: one of attributes of events or nodes, and the other of attributes of the edges or links.

Usage

```
netCoin(nodes = NULL, links = NULL, tree = NULL,
        community = NULL, layout = NULL,
        name = NULL, label = NULL, group = NULL, groupText = FALSE,
        labelSize = NULL, size = NULL, color = NULL, shape = NULL,
        border = NULL, legend = NULL, sort = NULL, decreasing = FALSE,
        ntext = NULL, info = NULL, image = NULL, imageNames = NULL,
        centrality = NULL,
        nodeBipolar = FALSE, nodeScaleLimits = NULL, nodeFilter = NULL,
        degreeFilter = NULL,
        lwidth = NULL, lweight = NULL, lcolor = NULL, ltext = NULL,
        intensity = NULL, linkBipolar = FALSE, linkScaleLimits = NULL,
        linkFilter = NULL,
        repulsion = 25, distance = 10, zoom = 1,
        fixed = showCoordinates, limits = NULL,
        main = NULL, note = NULL, showCoordinates = FALSE, showArrows = FALSE,
        showLegend = TRUE, frequencies = FALSE, statistics = FALSE,
        showAxes = FALSE, axesLabels = NULL,
        nodesLabel = NULL, linksLabel = NULL, scenarios = NULL, help = NULL,
        helpOn = FALSE, mode = c("network","heatmap"), roundedItems = FALSE,
```

```
controls = 1:8, cex = 1, background = NULL, defaultColor = "#1f77b4",
language = c("en", "es", "ca"), dir = NULL)
```

Arguments

nodes	a data frame with at least one vector of names.
links	a data frame with at least two vectors with source and target, including names of nodes.
tree	a data frame with two vectors: source and target, describing relationships between nodes.
name	name of the vector with names in the nodes data frame. By default, if language="en", name is "name".
label	name of the vector with labels in the nodes data frame.
group	name of the vector with groups in the nodes data frame.
groupText	show names of the groups.
community	algorithm to make communities: edge_betweenness("ed"), fast_greedy("fa"), label_prop("la"), leiden_eigen("le"), louvain("lo"), optimal("op"), spinglass("sp"), walktrap("wa")
centrality	calculates the centrality measures of a network. See calCentr .
labelSize	name of the vector with label size in the nodes data frame.
size	name of the vector with size in the nodes data frame.
color	name of the vector with color variable in the nodes data frame.
shape	name of the vector with shape variable in the nodes data frame.
border	name of the column with border width in the nodes data frame.
legend	name of the vector with the variable to represent as a legend in the nodes data frame.
ntext	name of the vector with html text in the nodes data frame.
info	name of the vector with information to display in a panel in the nodes data frame.
sort	name of the vector with node order in the nodes data frame (only for heatmap).
decreasing	decreasing or increasing sort of the nodes (only for heatmap).
intensity	name of the vector with intensity variable in the links data frame (only for heatmap).
lwidth	name of the vector with width variable in the links data frame.
lweight	name of the vector with weight variable in the links data frame.
lcolor	name of the vector with color variable in the links data frame.
ltext	name of the vector with labels in the links data frame.
nodeFilter	condition for filtering nodes.
linkFilter	condition for filtering links.
degreeFilter	numeric vector to filter the resulting network by degree.
nodeBipolar	a logical value that polarizes negative and positive node values in the graphical representation. Default = FALSE.

nodeScaleLimits	vector indicating the node scale limits, must be a numeric vector of length 2 or 3.
linkBipolar	a logical value that polarizes negative and positive link values in the graphical representation. Default = FALSE.
linkScaleLimits	vector indicating the link scale limits, must be a numeric vector of length 2 or 3.
defaultColor	a character vector giving a valid html color.
repulsion	a percentage for repulsion between nodes.
distance	a percentage for distance of links.
zoom	a number between 0.1 and 10 to start displaying zoom.
fixed	prevent nodes from being dragged.
scenarios	a note showing number of scenarios.
main	upper title of the graph.
note	lower title of the graph.
frequencies	a logical value true if the frequencies can be shown in barplots. Default = FALSE.
statistics	a logical value true if the possibility of calculating statistics with the nodes will be added to the final graph.
help	help text of the graph.
helpOn	Should the help be shown at the beginning?
background	background color or image of the graph.
layout	a matrix with two columns or an algorithm to elaborate the coordinates: davidson.harel.drl("da"), circle("ci"), Force-Atlas-2("fo"), fruchterman.reingold("fr"), gem("ge"), grid("gr"), kamada.kawai("ka"), lgl("lg"), mds("md"), random("ra"), reingold.tilford("re"), star("sta"), sugiyama("sug"). Several layouts can be given as a vector of algorithm names, as in c("fo", "fr", "ka"), or as a list mixing names and matrices; each becomes a plane of the plot, named as given in the list or after its algorithm.
limits	vector indicating size references to display layout, must be a numeric vector of length 4: x1, y1, x2, y2.
cex	number indicating the amount by which plotting text should be scaled relative to the default. Default = 1.
roundedItems	Display items with rounded borders.
controls	a numeric vector indicating which controls will be shown. 1 = sidebar, 2 = selection buttons, 3 = export buttons, 4 = nodes table, 5 = links table, 6 = search, 7 = zoom, 8 = legend. NULL hide all controls, negative values deny each control and 0 deny all.
mode	a character vector indicating the graph mode allowed: network, heatmap or both (both by default).
showCoordinates	a logical value true if the coordinates are to be shown in tables and axes. Default = FALSE.

showArrows	a logical value true if the directional arrows are to be shown. Default = FALSE.
showLegend	a logical value true if the legend is to be shown. Default = TRUE.
showAxes	a logical value true if the axes are to be shown. Default = FALSE.
axesLabels	a character vector giving the axes names.
nodesLabel	How should the nodes in this network be named?
linksLabel	How should the links in this network be named?
language	a character vector (es=spanish; en=english; ca=catalan).
image	name of the vector with image files in the nodes data frame.
imageNames	name of the vector with names for image files in the nodes data frame.
dir	a "character" string representing the directory where the web files will be saved.

Value

This function returns a netCoin object. If the 'dir' attribute is specified, the function creates a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Uribe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C) # edge data frame
netCoin(N, E) # netCoin object
```

netCorr

Networked correlations.

Description

netCorr produces a network object of correlations. Its input has to be at least one set of quantitative variables.

Usage

```
netCorr(variables, weight=NULL, pairwise=FALSE,
        minimum=-Inf, maximum=Inf, sort=FALSE, decreasing=TRUE,
        frequency=FALSE, means=TRUE,
        method=c("pearson", "kendall", "spearman"),
        criteria="p", Bonferroni=FALSE, minL=0, maxL=Inf,
        sortL=NULL, decreasingL=TRUE,
        igraph=FALSE, ...)
```

Arguments

variables	a data frame with at least two quantitative variables.
weight	a vector of weights. Optimal for data.frame tables
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.
minimum	minimum frequency to be considered
maximum	maximum frequency to be considered
sort	sort the correlation matrix according to the frequency of the events
decreasing	decreasing or increasing sort of the matrix
frequency	a logical value true if frequencies are to be shown. Default=FALSE.
means	a logical value true if means are to be shown. Default=TRUE.
method	a vector of statistics of similarity. Pearson correlation by default. spearman and kendall are also possible
criteria	statistic to be use for selection criteria.
Bonferroni	Bonferroni criterium of the signification test.
minL	minimum value of the statistic to include the edge in the list.
maxL	maximum value of the statistic to include the edge in the list.
sortL	sort the list according to the values of a statistic. See below
decreasingL	order in a decreasing way.
igraph	Produces an igraph object instead of a netCoin object if TRUE
...	Any netCoin argument.

Value

The function creates a netCoin object and eventually a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
data(iris)
netCorr(iris[,1:4],ltext="value",
  main="Correlations between measurements of Iris Species",
  note="Anderson, Edgar (1935) y Fisher, R. A. (1936)") # network object
```

netExhibit	<i>Images in a grid gallery.</i>
------------	----------------------------------

Description

netExhibit produces an interactive image gallery (alternative display).

Usage

```
netExhibit(tree, initialType = NULL, tableformat = FALSE, ...)
```

Arguments

tree	a data frame with two columns: source and target, describing relationships between nodes. It indicates a hierarchy between nodes which can be dynamically explored. Optionally, another two columns describing types can be passed.
initialType	A character vector indicating which node type will be shown at start.
tableformat	If the tree is specified as a table, one column per level.
...	Any exhibit argument.

Value

Object of class treeGallery_rd3.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# Create some film data
tree <- data.frame(
  film = c("The Shawshank Redemption", "The Godfather", "The Dark Knight", "The Godfather Part II",
    "12 Angry Men", "Schindler's List", "The Lord of the Rings: The Return of the King",
    "Pulp Fiction", "The Lord of the Rings: The Fellowship of the Ring",
    "The Good, the Bad and the Ugly"),
  director = c("Frank Darabont", "Francis Ford Coppola", "Christopher Nolan",
    "Francis Ford Coppola", "Sidney Lumet", "Steven Spielberg", "Peter Jackson",
    "Quentin Tarantino", "Peter Jackson", "Sergio Leone"),
  actor = c("Tim Robbins|Morgan Freeman|Bob Gunton", "Marlon Brando|Al Pacino|James Caan",
    "Christian Bale|Heath Ledger|Aaron Eckhart", "Al Pacino|Robert De Niro|Robert Duvall"),
```

```

      "Henry Fonda|Lee J. Cobb|Martin Balsam", "Liam Neeson|Ralph Fiennes|Ben Kingsley",
      "Elijah Wood|Viggo Mortensen|Ian McKellen", "John Travolta|Uma Thurman|Samuel L. Jackson",
      "Elijah Wood|Viggo Mortensen|Ian McKellen", "Clint Eastwood|Eli Wallach|Lee Van Cleef")
    )

    gallery <- netExhibit(tree, initialType="film", tableformat=TRUE)

    ## Not run:
    plot(gallery)

    ## End(Not run)

```

netGallery	<i>Images in a grid gallery.</i>
------------	----------------------------------

Description

netGallery produces an interactive image gallery.

Usage

```
netGallery(tree, deep = FALSE, initialType = NULL, tableformat = FALSE, ...)
```

Arguments

tree	a data frame with two columns: source and target, describing relationships between nodes. It indicates a hierarchy between nodes which can be dynamically explored. Optionally, another two columns describing types can be passed.
deep	The tree is specified in a structure that preserves paths. Each column will be a tree level, colnames as node types.
initialType	A character vector indicating which node type will be shown at start. No effects in 'deep' mode.
tableformat	If the tree is specified as a table, one column per level. No effects in 'deep' mode.
...	Any gallery argument.

Value

Object of class netGallery.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```

# simply linked tree
tree <- data.frame(
  parent=c("a","a","b","b","b","ab","ab","ab","ab","ba","ba","baa","ba"),
  child=c("aa","ab","ba","bb","bc","aba","abb","abc","abd","baa","bab","baaa","aba")
)

gallery <- netGallery(tree)

## Not run:
plot(gallery)

## End(Not run)

# Create some film data
tree <- data.frame(
  film = c("The Shawshank Redemption", "The Godfather", "The Dark Knight", "The Godfather Part II",
    "12 Angry Men", "Schindler's List", "The Lord of the Rings: The Return of the King",
    "Pulp Fiction", "The Lord of the Rings: The Fellowship of the Ring",
    "The Good, the Bad and the Ugly"),
  director = c("Frank Darabont", "Francis Ford Coppola", "Christopher Nolan",
    "Francis Ford Coppola", "Sidney Lumet", "Steven Spielberg", "Peter Jackson",
    "Quentin Tarantino", "Peter Jackson", "Sergio Leone"),
  actor = c("Tim Robbins|Morgan Freeman|Bob Gunton", "Marlon Brando|Al Pacino|James Caan",
    "Christian Bale|Heath Ledger|Aaron Eckhart", "Al Pacino|Robert De Niro|Robert Duvall",
    "Henry Fonda|Lee J. Cobb|Martin Balsam", "Liam Neeson|Ralph Fiennes|Ben Kingsley",
    "Elijah Wood|Viggo Mortensen|Ian McKellen", "John Travolta|Uma Thurman|Samuel L. Jackson",
    "Elijah Wood|Viggo Mortensen|Ian McKellen", "Clint Eastwood|Eli Wallach|Lee Van Cleef")
)

gallery <- netGallery(tree, initialType="film", tableformat=TRUE, color="type", scale=2)

## Not run:
plot(gallery)

## End(Not run)

# example with path preservation
tree <- rbind(
  c("a1","b1","c1|c2"),
  c("a2","b2","c3|c4"),
  c("a3","b2","c4|c5"),
  c("a4","b3|b4","c6|c7|c8")
)

colnames(tree) <- c("a","b","c")

gallery<-netGallery(tree,deep=TRUE,color="type")
## Not run:
plot(gallery)

## End(Not run)

```

pathCoin

*Structural Equation Models Graphs.***Description**

pathCoin produces a netCoin object from a lavaan object, i.e., parameters of structural equation model.

Usage

```
pathCoin(model, estimates=c("b", "se", "z", "pvalue", "beta"),
         fitMeasures=c("chisq", "cfi", "rmsea"), ...)
```

Arguments

model	a lavaan object.
estimates	A vector with at least one element amongst "b", "se", "z", "pvalue", "beta".
fitMeasures	Default values: "chisq", "df", "pvalue", "cfi", "rmsea"
...	Any netCoin argument.

Value

The function creates a netCoin object and eventually a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# Classic Wheaton et al. model
library(lavaan)
lower <- '
11.834
6.947 9.364
6.819 5.091 12.532
4.783 5.028 7.495 9.986
-3.839 -3.889 -3.841 -3.625 9.610
-21.899 -18.831 -21.748 -18.775 35.522 450.288 '
wheaton.cov <- lav_getcov(lower,
                          names = c("anomia67", "powerless67", "anomia71", "powerless71",
                                    "education", "sei"))

wheaton.model <- '
# latent variables
ses =~ education + sei
alien67 =~ anomia67 + powerless67
```

```

alien71 =~ anomia71 + powerless71
# regressions
alien71 ~ alien67 + ses
alien67 ~ ses
# correlated residuals
anomia67 ~~ anomia71
powerless67 ~~ powerless71
'

fit <- sem(wheaton.model, sample.cov = wheaton.cov, sample.nobs = 932)

pathCoin(fit)

```

pieCoin

pie charts.

Description

It generates pie charts from a coin object.

Usage

```

pieCoin(x, colors = c("#000000", "#8dc7e6", "#ffffff", "#005587"),
        nodes = NULL, links = NULL, name = NULL, lcolor = NULL, expected = TRUE,
        abline = NULL, main = NULL, note = NULL, showLegend = TRUE, help = NULL,
        helpOn = FALSE, cex = 1, language = c("en", "es", "ca"), dir = NULL)

```

Arguments

x	a coin object.
nodes	a data frame with information for each event.
links	a data frame with information for each pie.
name	name of the column with names in the nodes data frame.
colors	a vector of colors to be used when filling the slices.
lcolor	name of the column with color variable in the links data frame.
expected	Should expected coincidences be displayed?
abline	adds one or more straight lines between pies.
main	upper title of the graph.
note	lower title of the graph.
showLegend	a logical value true if the legend is to be shown.
help	a character string indicating a help text of the graph.
helpOn	Should the help be shown at the beginning?
cex	number indicating the amount by which plotting text should be scaled relative to the default.
language	a character vector (es=spanish; en=english; ca=catalan).
dir	a "character" string representing the directory where the web files will be saved.

Value

a pieCoin object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Hair by Eye by Sex table from M. Friendly (2000)
data(HairEyeColor)
H<-as.data.frame(HairEyeColor)
W<-H$Freq
I<-dichotomize(H,c("Hair","Eye","Sex"),add=FALSE)
C <- coin(I,w=W)
pie <- pieCoin(C)
## Not run:
plot(pie)

## End(Not run)
```

propCoin

Express Coin Entries as Fraction of Marginal Table

Description

This is like 'prop.table' for 'coin' objects.

Usage

```
propCoin(x, margin= 0, decimals=1)
```

Arguments

x	'coin' object.
margin	index, or vector of indices to generate margin for.
decimals	integer indicating the number of decimal places to be used.

Value

Table like 'x' expressed relative to 'margin'.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Random incidence matrix: 25 scenarios, 4 events.
I <- matrix(rbinom(100, 1, .5), nrow = 25, ncol = 4,
            dimnames = list(NULL, c("A", "B", "C", "D")))
C <- coin(I, sort = TRUE)

propCoin(C, 1)
```

renderLinks	Create an html list of links.
-------------	-------------------------------

Description

Create an html list of links.

Usage

```
renderLinks(data, columns, labels = NULL, target = "_blank", title = NULL,
            parallels = FALSE, sites = NULL)
```

Arguments

data	a data frame which contains the data.
columns	a column name which contains the urls.
labels	a column name which contains the text to display.
target	the target attribute specifies where to open the linked document: <code>'_blank'</code> opens the linked document in a new window or tab; <code>'_self'</code> opens the linked document in the same frame as it was clicked; <i>framename</i> opens the linked document in the named iframe.
title	a title displayed prior to the links.
parallels	should the links be displayed in two columns?
sites	a data frame of 3 columns (url, name, icon) with the sites that the function will recognize.

Value

a character vector of html formatted links.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
links <- data.frame(name = c(
  "Camarhynchus psitticula",
  "Camarhynchus pauper",
  "Camarhynchus parvulus"
), wikipedia=c(
  "https://en.wikipedia.org/wiki/Large_tree_finch",
  "https://en.wikipedia.org/wiki/Medium_tree_finch",
  "https://en.wikipedia.org/wiki/Small_tree_finch"
), wikidata=c(
  "https://www.wikidata.org/wiki/Q578835",
  "https://www.wikidata.org/wiki/Q1125857",
  "https://www.wikidata.org/wiki/Q1086136"
))
html <- renderLinks(links,c("wikipedia","wikidata"))
```

replaceClusters

Replace cluster columns in a netCoin object from surScat.

Description

replaceClusters replaces cluster columns in a netCoin object from surScat. Removes all existing cluster columns and adds the new one.

Usage

```
replaceClusters(scatObj, clusters, name=NULL, sort=TRUE,
  weight=NULL, sourceData=NULL, which=NULL, maxGroups=25)
```

Arguments

scatObj	A 'netCoin' object.
clusters	vector, factor, data.frame holding one clusterization per column, or clustering object (kmeans, poLCA, tidyLPA, looking4clusters).
name	name of the new cluster column. 'clusters' by default. When several clusterizations are added at once, each column keeps its own name, and 'name' becomes a prefix.
sort	sort clusters if requested (by mean coordinate on first layout axis).
weight	a vector of weights.
sourceData	optional data.frame for validation that clusters are in correct order.
which	clusterization to take when 'clusters' is a 'looking4clusters' object, given as a name or an index. Methods run for several numbers of groups yield their optimal solution, unless a specific number is asked for with the "name:k" form, as in which="kmeans:4". When it is left empty, every clusterization held by the object is added as its own column, one per number of groups, named after the method and that number, as in "kmeans(4)".

maxGroups clusterizations with more groups than this are skipped with a warning, which keeps a continuous variable passed in by mistake from becoming a column of hundreds of groups. NULL adds them all.

Details

The columns to remove are the ones surScat and [addClusters](#) recorded as holding clusterizations, so variables of the study are never removed, whatever they are named. Objects built before this was recorded fall back to the name of the column surScat creates.

The entries of \$clusters (see [surScat](#) and [addClusters](#)) belonging to the removed clusterizations are dropped along with their columns, and the new clusterization gets an entry of its own, its groups' centroids computed on every plane the object holds.

Value

A 'netCoin' object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

saveGhml	<i>Save a netCoin object as a .graphml file to be read in Gephi, Pajek, ...</i>
----------	---

Description

saveGhml produces a .graphml file from a netCoin object.

Usage

```
saveGhml(net, file="netCoin.graphml")
```

Arguments

net	A netCoin object.
file	The name of the file. If not extension, .gexf is used as default.

Value

The function creates a file with vertices and arcs or edges of a netCoin object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", add=FALSE, sep = "; ")
graph <- allNet(data, maxL=.5) # graph from an incidence matrix

saveGhml(graph,"graph") # save graph.net file

## End(Not run)
```

savePajek	<i>Save a netCoin object as a .net (.paj) file to be read in Pajek, Gephi, ...</i>
-----------	--

Description

savePajek produces a .net (.paj) file from a netCoin object.

Usage

```
savePajek(net, file="file.net", arcs=NULL, edges=NULL,
           partitions=NULL, vectors=NULL)
```

Arguments

net	a netCoin object.
file	The name of the file without extension. It will be .net or .paj according to data. The default is file.net or file.paj
arcs	Names of netCoin\$links to be included and considered as arcs in the Pajek file..
edges	Names of netCoin\$links to be included and considered as edges in the Pajek file..
partitions	Names of netCoin\$nodes to be included and considered as partitions in the Pajek file.
vectors	Names of netCoin\$nodes to be included and considered as vectors in the Pajek file.

Value

The function creates a file with vertices and arcs or edges of a netCoin object. Vectors and partitions can be also included. .

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", add=FALSE, sep = "; ")
graph <- allNet(data) # graph from an incidence matrix

savePajek(graph,"graph",edges="Haberman") # save graph.net file

## End(Not run)
```

shinyCoin

Include netCoin Plots in Shiny.

Description

Load a netCoin plot to display in shiny.

Usage

```
shinyCoin(x)
```

Arguments

x is a netCoin, barCoin or timeCoin object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
## Not run:
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep="; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C) # edge data frame
net <- netCoin(N, E) # netCoin object
shinyCoin(net)

## End(Not run)
```

sim	<i>Similarity matrix.</i>
-----	---------------------------

Description

It calculates a similarity/distance matrix from either an incidence data frame/matrix or a coin object.

Usage

```
sim(input, procedures="Jaccard", level=.95, distance=FALSE,
    minimum=1, maximum=Inf, sort=FALSE, decreasing=FALSE,
    weight = NULL, pairwise = FALSE)
```

Arguments

input	a binary data frame or a coin object, let's say an R list composed by a number of scenarios (\$n) and a coincidence matrix with frequencies (\$f).
procedures	a vector of statistics of similarity. See details below.
level	confidence level
distance	convert the similarity matrix into a distance matrix
minimum	minimum frequency to obtain a similarity/distance measure.
maximum	maximum frequency to obtain a similarity/distance measure.
sort	sort the list according to the values of a statistic. See details below
decreasing	order in a decreasing way.
weight	a vector of weights. Optimal for data.frame tables
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.

Details

Possible measures in procedures are

- Frequencies (f), Relative frequencies (x), Conditional frequencies (i), Coincidence degree (cc), Probable degree (cp),
- Expected (e), Confidence interval (con)
- Matching (m), Rogers & Tanimoto (t), Gower (g), Sneath (s), Anderberg (and),
- Jaccard (j), Dice (d), antiDice (a), Ochiai (o), Kulczynski (k),
- Hamann (ham), Yule (y), Pearson (p), odds ratio (od), Rusell (r),
- Haberman (h), Z value of Haberman (z).
- Hypergeometric p greater value (hyp).

Value

A similarity/distance matrix.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# From a random incidence matrix I(25X4)
I<-matrix(rbinom(100,1,.5),nrow=25,ncol=4,
  dimnames=list(NULL,c("A","B","C","D")))
sim(I)
#Same results
C<-coin(I)
sim(C)
```

sociologists

Data: Classical sociologists.

Description

Data frame with names, birth and death year data, birth country and movement.

Usage

```
data("sociologists")
```

Format

A data frame with life's period of 16 sociologists and the following 11 variables to study time coincidences:

name : name and last name of the sociologist.

birth : birth year.

death : death year.

birth_place : birth place.

birth_country : birth country.

death_place : death place.

death_country : death country.

label : combination of name, birth and death dates.

generation : generation (every 25 years) of the sociologists.

school : school of thought.

picture : name of the file where their picture is.

Source

Own elaboration from manuals of sociology.

References

See [events](#).

Examples

```
data(sociologists)
head(sociologists, 10)
tail(sociologists, 10)
```

surCoin	<i>Networked coincidences from a data frame.</i>
---------	--

Description

surCoin produces a network object of coincidences from a data frame converting variables into dichotomies.

Usage

```
surCoin(data, variables=names(data), commonlabel=NULL,
        dichotomies=NULL, valueDicho=1, metric=NULL, exogenous=NULL,
        weight=NULL, subsample=FALSE, pairwise=FALSE,
        minimum=1, maximum=nrow(data), sort=FALSE, decreasing=TRUE,
        frequency=FALSE, percentages=TRUE,
        procedures="Haberman", criteria="Z", Bonferroni=FALSE,
        support=-Inf, minL=-Inf, maxL=Inf,
        directed=FALSE, diagonal=FALSE, sortL=NULL, decreasingL=TRUE,
        igraph=FALSE, coin=FALSE, dir=NULL, ...)
```

Arguments

data	a data frame.
variables	a vector of variables included in the previous data frame.
commonlabel	a vector of variables whose names are to be included in nodes labels.
dichotomies	a vector of dichotomous variables to appear as just one category.
valueDicho	value or values to be selected for dichotomous variables. Default is 1.
metric	a vector of metrics.
exogenous	a vector of variables whose relations amongst them are of no interest. None by default.
weight	a vector of weights. Optimal for data.frame tables.
subsample	retract the analysis to scenarios with at least one event.
pairwise	Pairwise mode of handling missing values if TRUE. Listwise by default.
minimum	minimum frequency to be considered.
maximum	maximum frequency to be considered.

sort	sort the coincidence matrix according to frequency of events.
decreasing	decreasing or increasing sort of the matrix.
frequency	a logical value true if frequencies are to be shown. Default=FALSE.
percentages	a logical value true if percentages are to be shown. Default=TRUE.
procedures	a vector of statistics of similarity. See below.
criteria	statistic to be use for selection criteria.
Bonferroni	Bonferroni criterium of the signification test.
support	minimum value of the frequency of the coincidence to be edged.
minL	minimum value of the statistic to include the edge in the list.
maxL	maximum value of the statistic to include the edge in the list. By default is +Inf, except if criteria="Z" or criteria="hyp", in which case it is .5. It is recommended to change it to .05 if data has been sampled.
directed	includes same edges only once.
diagonal	includes auto-links.
sortL	sort the list according to the values of a statistic. See below.
decreasingL	order in a decreasing way.
igraph	Produces an igraph object instead of a netCoin object if TRUE.
coin	Only return the coincidences matrix if TRUE.
dir	a "character" string representing the directory where the web files will be saved.
...	Any netCoin argument.

Details

Possible measures in procedures are

- Frequencies (f), Relative frequencies (x), Conditional frequencies (i), Coincidence degree (cc), Probable degree (cp),
- Expected (e), Confidence interval (con)
- Matching (m), Rogers & Tanimoto (t), Gower (g), Sneath (s), Anderberg (and),
- Jaccard (j), Dice (d), antiDice (a), Ochiai (o), Kulczynski (k),
- Hamann (ham), Yule (y), Pearson (p), odds ratio (od), Rusell (r),
- Haberman (h), Z value of Haberman (z),
- Hypergeometric p greater value (hyp).
- Convert a matrix into an edge list (shape).

Value

This function creates a netCoin object (or igraph) and, if stated, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:10.18637/jss.v093.i11.

Examples

```
# A data frame with two variables Gender and Opinion
frame <- data.frame(Gender=c(rep("Man",3),rep("Woman",3)),
                    Opinion=c("Yes","Yes","No","No","No","Yes"))
surCoin(frame,commonlabel="") # network object

# A data frame with two variables (Gender and Hand) and nodes
input <- data.frame(
  Gender = c("Women", "Men", "Men", "Women", "Women","Men",
            "Men", "Men", "Women", "Women", "Men", "Women"),
  Hand    = c("Right", "Left","Right", "Right", "Right", "Right",
            "Left", "Right", "Right", "Left","Right", "Right"))
nodes <- data.frame(
  name = c("Gender:Men", "Gender:Women", "Hand:Left", "Hand:Right"),
  label = c("Women(50\u25)", "Men(50\u25)",
            "Left hand(25\u25)", "Right hand(75\u25)"))
G <- surCoin(input, nodes=nodes, proc=c("h","i"), label="label",
            ltext="i", showArrows=TRUE, maxL=.99)
```

surScat

Interactive scattergram of the cases of a data frame

Description

surScat produces an interactive scattergram of the cases (rows) of a data frame. Its axes are, by default, the two first dimensions of a multiple correspondence analysis (MCA) of the active qualitative variables, or of a principal component analysis (PCA) of the active quantitative variables, and its cases are colored by groups obtained by k-means clustering of these coordinates. Either axis may hold instead another factorial dimension or an observed variable (see xaxis and yaxis), and further planes and clusterizations can be added from [looking4clusters](#) (see layouts and clusters). Details explain how the axes are labelled, how each node is described and which group centroids the resulting object keeps.

Usage

```
surScat(data, variables=names(data), active=variables, columns=NULL,
        patterns=FALSE, vPatterns=variables, weight=NULL, seed=2020, maxN=2000,
        jitter=0, type=c("mca", "pca"), xaxis=NULL, yaxis=NULL, scaleAxes=TRUE,
        nfactors=2, critFactors=0, layouts=NULL, suffix=NULL, alignTo=NULL,
```

```
nclusters=2, clusterOn=c("factors", "variables"), scaleClusters=NULL,
nstart=25, clusterIndex=c("ch", "silhouette", "bic"), sortClusters=TRUE,
clusters=NULL, force_execution=FALSE, ...)
```

Arguments

data	a data frame.
variables	a vector of variables included in the previous data frame. Only the cases complete over all of them (and over weight) are analysed, and it defaults to every column of the data frame, which on a survey with filtered questions may leave few cases or none: it is therefore best stated. When less than half the cases are complete, a warning names the variables missing the most; when fewer than two are, it is an error.
active	a vector of active variables used to compute the coordinates.
columns	order of the columns of the resulting node data frame. NULL (default) orders them as the vPatterns variables (only when vPatterns has been explicitly stated, since it defaults to every variable), the n column of pattern counts, the active variables, the remaining ones (n is placed right after the active variables instead when the vPatterns block is empty), the k-means group columns, the ntext column and the name column, after which netCoin appends the fx and fy coordinates of the scattergram. A character vector places the columns it names first, in the stated order, and keeps every remaining column behind them in the default order, so no column is ever dropped. Such a vector may also contain the reserved names <code>_vpattern</code> , <code>_actives</code> , <code>_others</code> and <code>_nclusters</code> , which are expanded, where they are stated, into the vPatterns variables, the active variables, the variables which are neither of them, and the k-means group columns respectively. A column named twice (by itself and within a block) is kept at its first position. Names which are not a column of the node data frame are ignored with a warning.
patterns	if TRUE, cases sharing the same response pattern in vPatterns are collapsed into a single node, whose size represents the number of supporting cases (or the sum of their weights): a column n is added to the nodes and used as size argument of netCoin (unless another size is stated). In addition, a column ntext with an HTML description of each pattern (the values of its vPatterns variables plus its number of cases) is generated and passed as ntext argument of netCoin , unless another ntext is stated. Coordinates, k-means clustering and the eigenvalue-based factor selection are always computed case by case, exactly as when patterns=FALSE; collapsing only affects how the resulting node table is built afterwards: each pattern is plotted at the (weighted) mean position of its cases, variables outside vPatterns take their (weighted) mean if quantitative or their (weighted) mode if qualitative, and the k-means group columns list every distinct group reached by the pattern's cases, separated by "I" (a single value if all its cases share the same group). Recommended for MCA with few qualitative variables, where cases pile up on a small number of distinct positions. Automatically set to TRUE if vPatterns is stated.
vPatterns	a subset of variables defining the response pattern used to collapse cases when patterns=TRUE (or is implied by stating vPatterns). Defaults to variables, i.e.

cases are collapsed only if they agree on every variable. Stating a subset collapses cases sharing a pattern on just those variables: the remaining variables (including active ones) and the k-means group columns are then summarized as described under patterns, and the ntext description is restricted to just the vPatterns variables.

weight	a numeric vector of case weights, or the name of a column of the data frame containing them (this column is then excluded from variables). Weights are used to compute the MCA or PCA coordinates (weighted category masses in MCA, weighted correlations in PCA), so the axes represent the weighted structure of the data. The k-means clustering is then computed on these weighted coordinates without further weighting. When patterns collapses cases (see patterns and vPatterns), weights are also used for the mean/mode summaries and for the case count of each pattern. NULL (default) performs an unweighted analysis.
seed	random seed for reproducible k-means clustering, row sampling and jitter, and also for the methods of looking4clusters (see layouts and clusters), each of which starts from it. The RNG state of the session is restored on exit. Use NULL to avoid setting a seed.
maxN	maximum number of nodes to draw. A scattergram of more nodes than this, whether cases or patterns, is capped by drawing maxN of them at random, reproducibly for a given seed. The node table then holds that sample and nothing else, so a clusterization or a pair of coordinates computed on the whole data no longer lines up with it: addClusters and addAxes report the mismatch rather than guess, and the way to add either is to compute it on the node table of the resulting object. Raise maxN above the number of cases, or patterns, to keep them all.
jitter	amount of random noise added to the coordinates before plotting, expressed as a fraction of the range of each axis (e.g. .02 = 2 percent; TRUE is a shortcut for .02). It only spreads the points visually: clusters and coordinates are computed on the exact positions. It can be combined with patterns to separate patterns differing only in non active variables, and it is particularly useful when an axis holds a discrete observed variable (see xaxis and yaxis), whose few distinct values pile the cases up into columns.
type	Factorial type: mca for qualitative active variables, pca for quantitative active variables. If not stated, it is chosen automatically: pca if all the active variables are quantitative, mca otherwise.
xaxis, yaxis	what is plotted on the horizontal and on the vertical axis. NULL (default) plots the first and the second factor respectively. A factorial dimension can be stated as a number or as a name such as "F3" or "PC2" (any of these prefixes is accepted whatever the type of analysis), so that yaxis=3 plots the first factor against the third one. The name of a variable of the data frame plots that observed variable instead: it must be quantitative or an ordered factor, since placing a nominal variable on an axis would impose an arbitrary order on its categories. Should a column of the data frame be named like a factorial axis, the column prevails. A variable stated here does not need to be included in variables, as it is added to them: it thus takes part in the complete-cases filter and appears in the node table as an illustrative variable. It may also be an active one, although a variable is

then plotted against factors it has itself helped to build. Optionally, either argument accepts a second element, TRUE or FALSE, stating whether that axis is to be standardized (see `scaleAxes`), as in `xaxis=c("age", TRUE)` or `yaxis=c("F1", FALSE)`; `c()` turns such a pair into a character vector, which is read back correctly, but `list("age", TRUE)` is accepted as well. Note that combining an observed axis with `patterns=TRUE` plots each pattern at the (weighted) mean of that variable over its cases, unless the variable is itself part of `vPatterns`.

<code>scaleAxes</code>	if TRUE (default), the plotted factorial axes are each expressed in standard-deviation units (mean 0, sd 1), equivalent to dividing by the square root of their eigenvalue. Both MCA and PCA coordinates already have mean 0, so this only rescales their spread, making the two axes comparable and independent of how much variance/inertia they capture. Points are not bounded to any fixed range: their spread only follows the usual properties of a standardized variable. FALSE keeps the raw MCA/PCA coordinates, whose scale grows with the eigenvalue of each axis. An axis holding an observed variable (see <code>xaxis</code> and <code>yaxis</code>) is not affected: it keeps its own units, which would otherwise be lost. Both defaults can be overridden axis by axis with the second element of <code>xaxis</code> and <code>yaxis</code> .
<code>nfactors</code>	when <code>clusterOn="factors"</code> , the maximum number of MCA/PCA factors considered for clustering (<code>critFactors</code> then selects how many of these are actually used, at least 1). It only affects clustering: the scattergram plot itself always uses the factors stated by <code>xaxis</code> and <code>yaxis</code> (the two first ones by default), regardless of <code>clusterOn</code> , <code>nfactors</code> or <code>critFactors</code> . Ignored (with a warning if set explicitly) when <code>clusterOn="variables"</code> .
<code>critFactors</code>	when <code>clusterOn="factors"</code> , a factor among the first <code>nfactors</code> is used for clustering only if its eigenvalue exceeds <code>critFactors</code> times the mean eigenvalue of the full factorial spectrum. This generalizes Kaiser's rule (eigenvalue above average) to MCA, whose eigenvalues are not on the same scale as PCA's: <code>critFactors=1</code> reproduces the classic Kaiser criterion in (unweighted or weighted) PCA, since the mean eigenvalue of a correlation matrix is always 1, while in MCA it adapts to the actual inertia scale of the analysis. The default, 0, applies no filtering, so all <code>nfactors</code> factors are used. At least one factor is always used. Ignored (with a warning if set explicitly) when <code>clusterOn="variables"</code> .
<code>layouts</code>	dimensionality reductions computed by looking4clusters and added as further planes: any of "pca", "tsne", "cmds" (or "mds"), "nmf" and "umap", in the order they are to be offered. The plane of "pca" is offered as "cPC", its axes <code>cPC1</code> and <code>cPC2</code> : it is computed on the covariances of the active variables as they come (or of the 0/1 indicators of qualitative ones), so it is told apart from the PCA plane <code>surScat</code> draws from the standardized variables. As <code>surScat</code> does with its own factors, the axes of "pca" and "cmds" are labelled with the percentage of variance (or inertia, over the positive eigenvalues for <code>cmds</code> , whose canberra distance is not euclidean) each one holds. TRUE asks for all of them. NULL (default) adds none. <code>surScat</code> draws no links, so the network layouts accepted by netCoin ("fo", "la"...) do not apply to it: they are ignored with a warning. <code>tsne</code> , <code>nmf</code> and <code>umap</code> need the packages <code>Rtsne</code> , <code>NMF</code> and <code>uwot</code> respectively; <code>nmf</code> only accepts non-negative data, so it suits the 0/1 indicators of qualitative variables, but not quantitative variables holding negative values. Unlike the factorial axes, those of <code>tsne</code> and <code>umap</code> have no meaning of their own: only the proximity between points can be read.

suffix	a string. Every plane added by layouts is also offered aligned by Procrustes rotation to the main plane, under its own name followed by this suffix, so that both its original and its aligned version can be seen (see layoutsAlign). NULL (default) adds no aligned plane.
alignTo	NULL (default), or the name or the index of the plane the planes of suffix are aligned to by Procrustes rotation, in place of the main plane, so that they are read in its orientation. The name is that of a plane as the selector shows it ("varimax 1-3", "umap"...) or of a rotation asked for in layouts ("varimax", "quartimax", "equamax"), which stands for the first pair of its components. A plane not found is warned of and the main plane is used. The plane aligned to is not repeated, and the main plane is aligned too.
nclusters	number of clusters for k-means (2 by default, or 0 when both xaxis and yaxis are variables of the data frame rather than factorial axes). It can be a vector, in which case one group variable is added per value, the last one being the main solution: the one that colors the graph and describes each case. It may also hold "auto", for the number of groups to be chosen by clusterIndex. On its own, nclusters="auto" weighs from 2 to 10 groups and keeps only the chosen solution; next to a range, as in nclusters=c("auto", 2:6), it weighs the numbers of that range, keeps a group variable for each of them and places the chosen one last, so that it becomes the main solution. Numbers of groups reaching the number of distinct cases are left out. The values of the index for every number of groups weighed are kept in the attribute "clusterIndex" of the resulting object (see Value). nclusters=0 makes no groups at all: the plane is drawn alone, every case takes the same color unless color states a variable, the description of each case gives only its place on the two axes, and the methods asked for by clusters are left out with a warning, as they take their numbers of groups from nclusters.
clusterOn	what k-means clusters on: "factors" (default) uses a selection of the MCA/PCA factorial coordinates (see nfactors and critFactors). "variables" uses the active variables instead (their dichotomized indicators for mca, their standardized values for pca), so groups reflect the full active variable space rather than only the extracted factors. nfactors and critFactors are ignored in this case, with a warning if they are explicitly set.
scaleClusters	whether the columns k-means clusters on are each standardized (mean 0, sd 1) beforehand, so that every one of them has the same say in the distances. NULL (default) decides it by what is being clustered: TRUE for factorial coordinates, and also for the active variables of a pca, which come standardized anyway; FALSE for the dichotomized indicators of an mca (clusterOn="variables"). TRUE and FALSE force it either way. Standardizing matters because k-means weighs each column by its own spread. On raw factorial coordinates a dominant first factor cuts the groups along itself alone, and they then read as bands on a plot whose axes are standardized (see scaleAxes), which is why TRUE suits them. Dichotomized indicators are the opposite case. Standardizing one held by a proportion p of the cases turns a "yes" into $\sqrt{(1-p)/p}$ standard deviations, so a category held by 2% of the sample weighs seven times more than a balanced one, and squared distances add up over categories: the groups end up singling out the few cases holding a rare category. Left raw, the squared distance between

two cases is the number of categories they differ in, which is the usual metric for qualitative data.

Note that FALSE on factorial coordinates reproduces the behaviour of versions previous to this argument.

nstart number of random starts of the k-means algorithm, 25 by default. k-means only converges to a local optimum, the one its starting centres lead to, so it is run this many times and the best of those runs is kept. A single start (nstart=1, the behaviour of versions previous to this argument) reaches a clearly worse grouping in about one out of five runs, without any sign of it. Raising it costs proportionally more computing time, which is nonetheless a small part of the whole call.

clusterIndex the index by which nclusters="auto" chooses the number of groups: the one with the highest value of "ch" or "silhouette", or with the lowest value of "bic". "ch" (default) is the Calinski-Harabasz index, the between-group variance over the within-group variance, each divided by its degrees of freedom; it comes straight out of k-means, so it adds no computing time. "silhouette" is the mean silhouette width of the cases (package cluster), which compares the distance of every case to its own group with its distance to the nearest other group; it needs every distance between cases, so beyond 2000 cases it is computed on a random sample of 2000 of them.

Both judge the groups on all the active variables, whatever space k-means clusters on (see clusterOn): standardized for type="pca", and as the 0/1 indicators of their categories for type="mca", where the squared distance between two cases is the number of categories in which they differ. That is, k-means still makes its groups on the factors by default, but the number of groups is the one that best separates the cases on everything that was measured, not only on the plane. Judged on the factors themselves, the indices would mislead: a few factors make a low-dimensional space which, short of clear-cut groups, can be cut ever finer for an ever better index, so that the largest number of groups weighed would nearly always be chosen; with qualitative variables this is worse still, as the cases pile up on a few points of the plane. A cut that only splits the plane gains nothing on the whole of the active variables. Mind that this judge tends, on the contrary, to prefer few groups, often 2, when the data hold no clear structure. With clusterOn="variables" k-means clusters and the indices judge on the same variables. Neither index takes case weights into account, as k-means does not either.

"bic" (or "BIC") is the Bayesian information criterion of the latent model that suits the active variables (see clusters): a latent profile analysis for type="pca", estimated by package mclust, or a latent class analysis for type="mca", estimated by package poLCA. It is the usual criterion for the number of classes of such models and it may also choose a single group. The number it chooses is the one of every method, the k-means of surScat included. Should the package be missing, or the model fail for every number of groups, Calinski-Harabasz is used instead, with a warning, being the index of k-means closest to it: both rest on the within-group variance, which governs the likelihood of a mixture of spherical normal distributions, of which k-means is the limiting case. Ignored unless nclusters holds "auto".

sortClusters	if TRUE (default), the clusters of each k-means solution are renumbered so that group 1 is the one whose (weighted) mean on the horizontal axis is the lowest, group 2 the next one, and so on. k-means numbers its clusters after its own initialization, which is arbitrary, so this is what gives a meaning to the order of the group columns, which are ordered factors. The axis it sorts on is the one actually plotted, that is, the first factor unless xaxis states an observed variable, so that the numbering always follows what the reader sees. It applies as well when clusterOn="variables". Only the numbering changes: the partition of the cases is exactly the same. FALSE keeps the numbering produced by k-means, as previous versions did for a given seed.
clusters	clustering methods whose groups are added as further columns: any of "pam", "hclust" and "kmeans" of looking4clusters , and the latent models "lpa" and "lca". TRUE asks for all of the former and for the latent model that suits the active variables, provided its package is installed: a latent profile analysis (package mclust, with equal variances and no covariances, model "EEI") for type="pca", or a latent class analysis (package poLCA, estimated 5 times over from random starts, the best being kept) for type="mca". Stated by name, a latent model that does not suit the active variables, or whose package is missing, is left out with a warning; so is a number of groups for which it cannot be estimated. Their columns are named in capitals, as in "LPA(3)" or "LCA(3)". The k-means of looking4clusters is not a copy of surScat's own: it works on the active variables as they come (see the Description), not on the standardized factorial coordinates, and it is started once, whereas surScat starts its own nstart times. Only the numbers of groups stated by nclusters are kept, in the same order, and each column is named after its method and its number of groups, as in "pam(3)". pam and hclust need the packages fpc, cluster and dendextend. NULL (default) adds none.
force_execution	looking4clusters leaves out pam, hclust and cmds when there are more than 5000 cases, since they need the whole matrix of distances between them, and says so with a warning. TRUE computes them anyway.
...	Any netCoin argument. color, in particular, states a column of the node table to color the cases by, in place of the main k-means solution, whose group columns are kept all the same.

Details

Axes: Either axis, or both, may be changed with xaxis and yaxis: a different factorial dimension, or an observed variable of the data frame. Clustering is not affected by this choice, so a plot of an observed variable against a factor still colors its cases according to the whole structure found in the active variables.

Axis labels: A plotted factorial axis is labelled with the percentage of variance (PCA) or of inertia (MCA) it accounts for, computed on the full factorial spectrum. In MCA the raw inertia percentages are known to be strongly pessimistic, so Benzecri's adjustment is applied: only the factors whose eigenvalue exceeds $1/Q$ (Q being the number of active variables) are retained, their eigenvalues are transformed into $(Q/(Q-1))^2$ (eigenvalue - $1/Q$)², and the percentages are computed on this adjusted inertia. The adjustment is skipped, and the raw inertia percentages are kept, when the second factor does not itself exceed the $1/Q$ threshold, since it would then be labelled

as 0 percent. An axis holding an observed variable is labelled with the label of that variable, or with its name if it has none. These labels are passed to [netCoin](#) as its `axesLabels` argument, unless another `axesLabels` is stated.

Node descriptions: Every node carries an HTML description, in a column named `ntext` which is passed to [netCoin](#) as its `ntext` argument, unless another `ntext` is stated. When cases are collapsed it describes the pattern (see [patterns](#)). Otherwise it describes the case itself: its value on each of the two axes, followed in brackets by the mean of the k-means group it belongs to on that same axis, and the number of that group (the last k-means solution, when `nclusters` states several). A factorial axis is described by the coordinate which is plotted, whereas an observed variable is described in its own units and with its own labels, even if its axis has been standardized.

Group centroids: The resulting object also holds `$clusters`, the (weighted) centroid of every group of every k-means solution, on the plane just drawn: a flat list with one matrix per clusterization, its name joining the clusterization and the plane with a dot, as in `S$clusters[["Groups(2).PCA"]]`, and one row per group. The plane [surScat](#) draws is named after the analysis ("PCA" or "MCA") when it holds the first two factors, and after its two axes otherwise (as in "PC1-PC3"). When the nodes are patterns whose cases fall in several groups (see [vPatterns](#)), each pattern takes part in the centroid of every group it spans with the (weighted) cases it holds of that group, as recorded in the attribute "clusterShares" (see [addClusters](#)). [addAxes](#) extends it with an entry for every clusterization on whichever plane it adds, and [addClusters](#) with an entry on every plane the object already holds for whichever clusterization it adds, so that `$clusters` always holds one entry per clusterization-plane pair. Printing it, on its own or as part of the object, only states how many clusterizations and planes it covers.

It also holds `$clusterMeans`, the (weighted) mean of every group of every clusterization on every numeric column of the node table but `fx` and `fy`: a list with one data frame per clusterization, as in `S$clusterMeans[["Groups(2)"]]`, with one row per group, in the same order as the rows of its entries in `$clusters`, and one column per variable. These are the centroids of the groups when the graph places a variable on an axis rather than a coordinate of a plane: a centroid being the mean on each axis separately, any two of its columns give the centroid on the plane they span, and either of them can also be paired with a column of an entry of `$clusters`. They are computed just as `$clusters`, patterns spanning several groups included, and kept up to date by [addClusters](#) and [replaceClusters](#).

Planes and clusterizations from [looking4clusters](#): Further planes and clusterizations can be obtained from [looking4clusters](#) with the arguments `layouts` and `clusters`. They are added to the plane and to the k-means groups that [surScat](#) draws, never put in their place: the main plane is still the one stated by `type`, `xaxis` and `yaxis`, and the selector of the graph offers the others next to it. [looking4clusters](#) works on the same cases as the k-means of [surScat](#), one row each, but on the active variables as they come: the quantitative ones unstandardized (so its "pca" differs from the PCA of [surScat](#), which standardizes them) and the qualitative ones as a matrix of 0/1 indicators. When the nodes are patterns, its results are collapsed as [addAxes](#) and [addClusters](#) collapse them; when `maxN` has drawn a sample, only the cases of the nodes drawn are kept. Case weights are not taken into account by [looking4clusters](#), whose methods accept none. The same holds for the latent models clusters may also ask for: a latent profile analysis (LPA) when the active variables are quantitative, and a latent class analysis (LCA) when they are qualitative, which take the active variables as they come too, and whose classes are added in the same way. Once more than one plane is held, each one keeps the titles of its own axes in the graph (as "F1 (23.5%)", "PC1" or "UMAP2", followed by suffix in an aligned plane), unless `axesLabels` is stated, which then serves for every plane.

Value

This function creates a netCoin object and, if stated, a folder in the computer with an HTML document named index.html which contains the produced graph. This file can be directly opened with your browser and sent to a web server to work properly.

When nclusters holds "auto", the object carries an attribute "clusterIndex": a data frame with one row per number of groups weighed, holding that number (k), the value of the index (in a column named after clusterIndex) and whether it was the one chosen (optimal).

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

References

Escobar, M. and Martinez-Urbe, L. (2020) Network Coincidence Analysis: The netCoin R Package. *Journal of Statistical Software*, **93**, 1-32. doi:[10.18637/jss.v093.i11](https://doi.org/10.18637/jss.v093.i11).

Examples

```
# A data frame with two variables Gender and Opinion
frame<-data.frame(Gender=c(rep("Man",3),rep("Woman",3)),
  Opinion=c("Yes","Yes","No","No","No","Yes"))
surScat(frame) # network object

# PCA example: quantitative active variables, clustered on those variables
# (not on the two plotted PCA axes), shaped by a non-active variable
data(iris)
surScat(iris, active=c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
  shape="Species", nclusters=2:3, clusterOn="variables")

# An observed variable on the horizontal axis, read against the second component,
# and the first component against the third one
surScat(iris, active=c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
  xaxis="Petal.Width", shape="Species")
surScat(iris, active=c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
  yaxis="PC3", shape="Species")

# The number of groups chosen by the Calinski-Harabasz index among 2 to 6, keeping
# every solution, and the values of the index
S <- surScat(iris, active=c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
  shape="Species", nclusters=c("auto", 2:6))
S
attr(S, "clusterIndex")

# Further planes and clusterizations from looking4clusters: the PCA of the raw (not
# standardized) variables and a UMAP, each also aligned to the main plane (suffix),
# and the kmeans, pam and hclust groups of looking4clusters for the same numbers of
# groups as the k-means of surScat

if(requireNamespace("uwot", quietly=TRUE) && requireNamespace("fpc", quietly=TRUE) &&
  requireNamespace("cluster", quietly=TRUE) && requireNamespace("dendextend", quietly=TRUE)) {
```

```

surScat(iris, active=c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
        shape="Species", nclusters=2:3, layouts=c("pca", "umap"), clusters=TRUE,
        suffix="")
}

# Qualitative active variables: looking4clusters works on their 0/1 indicators,
# which suit nmf. clusters may also name only some of its methods.
data(ess)
if(requireNamespace("Rtsne", quietly=TRUE) && requireNamespace("NMF", quietly=TRUE) &&
    requireNamespace("fpc", quietly=TRUE) && requireNamespace("cluster", quietly=TRUE) &&
    requireNamespace("dendextend", quietly=TRUE)) {
  surScat(ess, variables=c("Gender", "Age", "Social participation",
                          "Political participation"),
          weight="cweight", patterns=TRUE, nclusters=3,
          layouts=c("tsne", "nmf", "mds"), clusters=c("hclust", "kmeans"), suffix="*")
}

# Latent profiles (LPA) of the quantitative active variables, next to the k-means
# groups, for 2 to 4 profiles
if(requireNamespace("mclust", quietly=TRUE)) {
  P <- surScat(iris, active=c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"),
              shape="Species", nclusters=2:4, clusters="lpa")

  P
  table(P$nodes$Species, P$nodes[["LPA(3)"]])
}

# Latent classes (LCA) of the qualitative active variables, their number chosen by the
# BIC of the latent class model among 1 to 4, which then sets that of k-means too
if(requireNamespace("poLCA", quietly=TRUE)) {
  L <- surScat(ess, variables=c("Gender", "Age", "Social participation",
                              "Political participation"),
              weight="cweight", patterns=TRUE, nclusters=c("auto", 1:4),
              clusterIndex="bic", clusters="lca")

  L
  attr(L, "clusterIndex")
}

```

timeCoin

Networked coincidences.

Description

timeCoin produces a timeCoin object.

Usage

```

timeCoin(periods, name = "name", start = "start", end = "end", group = NULL,
         text = NULL, main = NULL, note = NULL, info = NULL,
         events = NULL, eventNames = "name", eventPeriod = "period",

```

```
eventTime = "date", eventColor = NULL, eventShape = NULL,
cex = 1, language = c("en", "es", "ca"), dir = NULL)
```

Arguments

periods	a data frame with at least three vectors of name, start and end of the periods.
name	name of the vector with names in the periods data frame.
start	name of the vector with starts in the periods data frame.
end	name of the vector with ends in the periods data frame.
group	name of the vector with groups in the periods data frame.
text	name of the vector with html text in the periods data frame.
main	upper title of the graph.
note	lower title of the graph.
info	name of the vector with information to display in a panel in the periods data frame.
events	a data frame of events included into the periods with three columns: event name, periodParent and eventTime
eventNames	name of the vector with names in the events data frame.
eventPeriod	name of the vector with period names in the events data frame.
eventTime	name of the vector with time points in the events data frame.
eventColor	name of the vector with color criteria in the events data frame.
eventShape	name of the vector with shape criteria in the events data frame.
cex	number indicating the amount by which plotting text should be scaled relative to the default. Default = 1.
language	a character vector (es=spanish; en=english; ca=catalan).
dir	a "character" string representing the directory where the web files will be saved.

Value

Object of class timeCoin.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# Database of the classical sociologists.
# Only periods
data(sociologists)
timeCoin(sociologists, "name", "birth", "death", "school")

# Periods and events
data(works)
timeCoin(sociologists, "name", "birth", "death", "school",
        events=works, eventNames="label", eventPeriod="author", eventTime="date")
```

toIgraph	<i>igraph object.</i>
----------	-----------------------

Description

igraph object from a netCoin object.

Usage

```
toIgraph(net)
```

Arguments

net is a netCoin object. See [netCoin](#)

Value

An igraph object.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Examples

```
# A character column (with separator)
frame <- data.frame(A = c("Man; Women", "Women; Women",
                          "Man; Man", "Undet.; Women; Man"))
data <- dichotomize(frame, "A", sep = "; ")[2:4]
C <- coin(data) # coincidence matrix
N <- asNodes(C) # node data frame
E <- edgeList(C)
net <- netCoin(N, E)
toIgraph(net) # conversion into a igraph object
```

works	<i>Data: Classical sociological works.</i>
-------	--

Description

Data frame with classical sociological works written by authors in the sociologists data frame.

Usage

```
data("sociologists")
```

Format

A data frame with 54 observations (events) and the following 4 variables to study coincidences in time:

name : name and last name of the author of the work.

label : abbreviation of the complete name.

works : work's name.

date : year of its first publication.

Author(s)

Modesto Escobar, Department of Sociology and Communication, University of Salamanca.

Source

Own elaboration from manuals of sociology.

References

See [events](#).

Examples

```
data(works)
head(works, 10)
tail(works, 10)
```

Index

* datasets

- dice, [39](#)
 - ess, [45](#)
 - events, [46](#)
 - families, [49](#)
 - finches, [50](#)
 - Galapagos, [52](#)
 - links, [71](#)
 - sociologists, [100](#)
 - works, [114](#)
- [addAxes](#), [6](#), [9](#), [14](#), [65](#), [67](#), [68](#), [70](#), [71](#), [105](#), [110](#)
- [addClusters](#), [7](#), [8](#), [11](#), [12](#), [14](#), [27](#), [29](#), [96](#), [105](#), [110](#)
- [addCommunities](#), [8](#), [10](#), [14](#)
- [addDescription](#), [12](#)
- [addImage](#), [13](#)
- [addNetCoin](#), [14](#)
- [addTutorial](#), [15](#)
- [allNet](#), [16](#)
- [asGallery](#), [18](#)
- [asNodes](#), [5](#), [19](#)
- [barCoin](#), [20](#)
- [calCentr](#), [21](#), [84](#)
- [caring](#), [23](#)
- [caring_create_graphs](#), [24](#)
- [caring_read_file](#), [25](#)
- [clusterAgree](#), [25](#), [28](#), [29](#)
- [clusterMatch](#), [26](#), [27](#), [27](#)
- [cobCoin](#), [29](#)
- [coefTable](#), [31](#), [61](#)
- [coexist](#), [33](#), [44](#)
- [coin](#), [4](#), [35](#), [49](#)
- [coocur](#), [36](#)
- [d_netCorr](#), [37](#)
- [dice](#), [39](#), [46](#)
- [dichotomize](#), [40](#)
- [distant](#), [42](#)
- [dyncohort](#), [34](#), [43](#)
- [edgeList](#), [5](#), [44](#)
- [ess](#), [45](#)
- [events](#), [39](#), [46](#), [101](#), [115](#)
- [exhibit](#), [47](#), [88](#)
- [expectedList](#), [48](#)
- [families](#), [49](#)
- [finches](#), [50](#)
- [fromIgraph](#), [51](#)
- [Galapagos](#), [52](#)
- [gallery](#), [53](#), [89](#)
- [gallery2 \(exhibit\)](#), [47](#)
- [gallery3](#), [55](#)
- [get_panel_template](#), [56](#)
- [get_template](#), [57](#)
- [get_template2](#), [58](#)
- [glmCoin](#), [31](#), [33](#), [59](#), [77–79](#)
- [glmCoin2](#), [61](#)
- [incTime](#), [62](#)
- [layoutCircle](#), [63](#)
- [layoutDist](#), [64](#), [67–71](#)
- [layoutGrid](#), [66](#)
- [layoutsAlign](#), [66](#), [70](#), [71](#), [107](#)
- [layoutsDist](#), [65–68](#), [69](#)
- [links](#), [71](#)
- [logCoin](#), [72](#)
- [looking4clusters](#), [74](#), [103](#), [105](#), [106](#), [109](#), [110](#)
- [lower](#), [75](#)
- [makeT](#), [76](#)
- [makeT2](#), [76](#)
- [marginsTable](#), [31](#), [33](#), [60](#), [77](#)
- [match.fun](#), [78](#)
- [mobileEdges](#), [80](#)

multigraphCreate, [80](#), [82](#)
multiPages, [82](#)

netCoin, [5](#), [14](#), [17](#), [18](#), [22](#), [30](#), [34](#), [38](#), [43](#), [51](#),
 [60](#), [62](#), [73](#), [83](#), [87](#), [91](#), [102](#), [104](#), [106](#),
 [109](#), [110](#), [114](#)
netCoin-package, [4](#)
netCorr, [12](#), [86](#)
netExhibit, [88](#)
netGallery, [89](#)
netGallery2 (netExhibit), [88](#)

pathCoin, [91](#)
pieCoin, [92](#)
print.coefTable (coefTable), [31](#)
print.marginsTable (marginsTable), [77](#)
propCoin, [93](#)

renderLinks, [94](#)
replaceClusters, [95](#), [110](#)

saveGhml, [96](#)
savePajek, [97](#)
shinyCoin, [98](#)
sim, [5](#), [42](#), [99](#)
sociologists, [100](#)
surCoin, [30](#), [101](#)
surScat, [7](#), [9–12](#), [14](#), [27](#), [29](#), [65](#), [68](#), [71](#), [96](#),
 [103](#)

timeCoin, [34](#), [44](#), [112](#)
toIgraph, [114](#)

works, [114](#)