Package ‘streamDAG’

October 6, 2023

Version 1.5

Date 2023-10-6

Title Analytical Methods for Stream DAGs

Maintainer Ken Aho <ahoken@isu.edu>

Depends R (>= 4.0), igraph

Imports asbio, graphics, stats, plotrix

VignetteBuilder knitr

Suggests ggplot2, sf, knitr, rmarkdown, bookdown, RColorBrewer, devtools, ggrepel

Description
Provides indices and tools for directed acyclic graphs (DAGs), particularly DAG representations of intermittent streams. A detailed introduction to the package can be found in the publication: ``Non-perennial stream networks as directed acyclic graphs: The R-package streamDAG'' (Aho et al., 2023) <doi:10.1016/j.envsoft.2023.105775>, and in the introductory package vignette.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Author Ken Aho [aut, cre],
Arya Legg [dtc, ctb],
Rob Ramos [dtc],
Maggi Kraft [dtc],
Charles T. Bond [dtc],
Rebecca L. Hale [dtc]

Repository CRAN

Date/Publication 2023-10-06 18:50:02 UTC

R topics documented:

A — *Arcs of a directed graph*

Description

This function and its documentation have been lifted from the *igraph* function `E` with arguments according to DAG conventions. An arc sequence is a vector containing numeric arc ids, with a special class attribute that allows custom operations: selecting subsets of arcs based on attributes, or graph structure, creating the intersection, union of arcs, etc.

Usage

```
A(G, P, path)
```

Arguments

`G`	Graph object of class `igraph`. See `graph_from_literal`.
`P`	A list of node to select arcs via pairs of nodes. The first and second nodes select the first arc, the third and fourth node select the second arc, etc.
`path`	A list of nodes, to select arcs along a path. Note that this only works reliably for simple graphs. If the graph has multiple arcs, one of them will be chosen arbitrarily to be included in the arc sequence.

Details

Arc sequences are usually used as function arguments that refer to arcs of a graph.

An arc sequence is tied to the graph it refers to: it really denoted the specific arcs of that graph, and cannot be used together with another graph.

An arc sequence is most often created by the `A()` function. The result includes arcs in increasing arc id order by default (if none of the `P` and `path` arguments are used). An arc sequence can be indexed by a numeric vector, just like a regular R vector.

Value

An arc sequence of the graph.

Author(s)

Gabor Csardi

See Also

See `E`

Examples

```
G <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e, e --+ j, j --+ m, f --+ g, g --+ i,
 h --+ i, i --+ k, k --+ l, l --+ m, m --+ n,  n --+ o)

A(G)
```

A.mult *Raise an adjacency matrix to some power*

Description

When applying the definition of matrix multiplication to an adjacency matrix $\mathbf{A}$, the i, j entry in $\mathbf{A}^k$ will give the number of paths in the graph from node i to node j of length k.

Usage

```
A.mult(G, power, text.summary = TRUE)
```

Arguments

`G`	Graph object of class `igraph`. See `graph_from_literal`.
`power`	The power to rise the adjacency matrix to.
`text.summary`	Logical. If `TRUE` the function returns a summary of the paths of length `power`. If `FALSE`. The adjacency matrix raised to `power` is returned.

Value

Returns either a character vector of paths of a specified length or, if `text.summary = TRUE`, the adjacency matrix raised to a specified power.

Author(s)

Ken Aho

Examples

```
kon_full <- streamDAGs("konza_full")
A.mult(kon_full, power = 6)
```

AIMS.node.coords *Nodal coordinates for graphs in the AIMS project*

Description

Contains spatial coordinates for graph nodes for stream networks in the Aquatic Intermittency effects on Microbiomes in Streams (AIMS) project

Usage

```
data("AIMS.node.coords")
```

Format

A data frame with 307 observations on the following 7 variables.

`Object.ID` Nodal identifier

`lat` Latitude

`long` Longitude

`site` Stream network name, currently includes: `"KZ"` = Konza Prairie, `"TD"` = Talladega, `"WH"` = Weyerhauser, `"PR"` = Painted Rock, `"JD"` = Johnson Draw, `"DC"` = Dry Creek, and `"GJ"` = Johnson Draw.

`piezo` Logical, indicating whether the location contains a peizometer.

`microbial_seasonal_network` Logical, whether the location was sampled as part of AIMS seasonal microbial sampling.

`STIC_inferred_PA` Logical, whether surface water presence/absence data were obtained from STIC (Stream Temperature, Intermittency, and Conductivity) sensors at the location.

arc.pa.from.nodes *Obtain arc stream activity outcomes based on bounding nodes*

Description

Given nodal water presence absence data $\in \{0, 1\}$ for a graph, G, the function calculates arc water presence probabilities using particular rules (see approaches in Details).

Usage

```
arc.pa.from.nodes(G, node.pa, approach = "aho", na.rm = TRUE)
```

Arguments

G	Graph object of class igraph. See graph_from_literal.
node.pa	A data frame or matrix of nodal presence absence data with column names corresponding to node names in G.
approach	One of "aho", "dstream", or "ustream" (see Details).
na.rm	For approach = "aho", one of TRUE or FALSE indicating whether NA values should be stripped before calculating means. Ignored for other appraches.

Details

The approach argument currently supports three alternatives "aho", "dstream" and "ustream". Let x_k represent the kth arc with bounding nodes u and v.

Under approach = "aho" there are three possibilities: $x_k = 1.0$ if both u and v are wet, $x_k = 0$ if both u and v are dry, and $x_k = 0.5$ if only one of u or v is wet.

Under approach = "dstream", $x_k = 1.0$ if v is wet, and $x_k = 0$ if v is dry.

Conversely, if approach = "ustream", $x_k = 1.0$ if *u* is wet, and $x_k = 0$ if *u* is dry.

Value

Returns a matrix whose entries are estimated probabilities of success (e.g. surface water presence) based on the rules given in Aho et al. (2023). Matrix columns specify arcs and rows typically represent time series observations.

Author(s)

Ken Aho

References

Aho, K., Derryberry, D., Godsey, S. E., Ramos, R., Warix, S., Zipper, S. (2023) The communication distance of non-perennial streams. EarthArXiv https://eartharxiv.org/repository/view/4907/

Examples

```
murphy_spring <- graph_from_literal(IN_N --+ M1984 --+ M1909, IN_S --+ M1993,
M1993 --+ M1951 --+ M1909 --+ M1799 --+ M1719 --+ M1653 --+ M1572 --+ M1452,
M1452 --+ M1377 --+ M1254 --+ M1166 --+ M1121 --+ M1036 --+ M918 --+ M823,
M823 --+ M759 --+ M716 --+ M624 --+ M523 --+ M454 --+ M380 --+ M233 --+ M153,
M153 --+ M91 --+ OUT)

data(mur_node_pres_abs)
pa <- mur_node_pres_abs[400:405,][,-1]
arc.pa.from.nodes(murphy_spring, pa)
arc.pa.from.nodes(murphy_spring, pa, "dstream")
```

assort *Assortativity*

Description

Calculates graph assortativity

Usage

```
assort(G, mode = "in.out")
```

Arguments

G	Graph object of class igraph. See graph_from_literal.
mode	One of "in.in", "in.out", "out.out", "out.in", or "all".

Details

The definitive measure of graph assortativity is the Pearson correlation coefficient of the degree of pairs of adjacent nodes (Newman, 2002). Let $\overrightarrow{u_i v_i}$ define nodes and directionality of the ith arc, $i = 1, 2, 3, \ldots, m$, let $\gamma, \tau \in -, +$ index the degree type: $- = in, + = out$, and let (u_i^γ, v_i^τ), be the $\gamma-$ and $\tau-$degree of the *i*th arc. Then, the general form of assortativity index is:

$$r\left(\gamma, \tau\right) = m^{-1} \frac{\sum_{i=1}^{m} (u_i^\gamma - \bar{u}^\gamma)(v_i^\tau - \bar{v}^\tau)}{s^\gamma s^\tau}$$

where $\bar{u}^\gamma$ and $\bar{v}^\gamma$ are the arithmetic means of the u_i^γs and v_i^τs, and s^γ and s^τ are the population standard deviations of the u_i^γs and v_i^τs. Under this framework, there are four possible forms to $r\left(\gamma, \tau\right)$ (Foster et al., 2010). These are: $r\left(+, -\right), r\left(-, +\right), r\left(-, -\right)$, and $r\left(+, +\right)$.

Value

Assortativity coefficeint outcome(s)

Author(s)

Ken Aho, Gabor Csardi wrote degree

References

Newman, M. E. (2002). Assortative mixing in networks. *Physical Review Letters, 89*(20), 208701.

Foster, J. G., Foster, D. V., Grassberger, P., & Paczuski, M. (2010). Edge direction and the structure of networks. *Proceedings of the National Academy of Sciences, 107*(24), 10815-10820.

Examples

```
network_a <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e, e --+ j,
j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l, l --+ m, m --+ n,
n --+ o)
assort(network_a)
```

---

bern.length *Botter and Durighetto Bernoulli stream length*

Description

A simple function for calculating the dot product of a vector of stream arc lengths and a corresponding vector of either binary (stream presence or absence) outcomes, probabilities of stream presence or inverse probabilites of stream presence.

Usage

```
bern.length(lengths, pa, mode = "local")
```

Arguments

`lengths`	A numeric vector of stream arc lengths
`pa`	A numeric vector of either binary (stream presence or absence) outcomes, probabilities of stream presence or inverse probabilites of stream presence. A vector outcome in `lengths` should correspond to an outcome for the same arc in `pa`.
`mode`	One of `"local"` of `"global"`

Value

When `pa` is a vector of binary (stream presence or absence) data, the function provides a measure of instantaneous stream length (in the units used in `lengths`). When `pa` is a vector of probabilities of stream presence, the function provides average stream length (in units used in `lengths`). When `pa` is a vector of inverse probabilites of stream presence, the function provides average communication distance (in units used in `lengths`).

Author(s)

Ken Aho

References

Botter, G., & Durighetto, N. (2020). The stream length duration curve: A tool for characterizing the time variability of the flowing stream length. *Water Resources Research, 56*(8), e2020WR027282.

Examples

```
lengths <- rexp(10, 10)
pa <- rbinom(10, 11, 0.4)
bern.length(lengths, pa)
```

beta.posterior *Posterior Beta and Inverse-beta summaries*

Description

Calculates summaries for beta and inverse-beta posteriors given prior probabilities for success, binary data and prior weight specification. Summaries include beta and inverse beta posterior means and variances and stream length and communication distance summaries given that stream `length` is provided for intermittent stream segments.

Usage

```
beta.posterior(p.prior, dat, length = NULL, w = 0.5)
```

Arguments

`p.prior`	Prior probability for success for the beta prior. The beta prior for the probability of success (e.g., stream presence) for *k*th outcome (e.g., stream segment) is defined as: $\theta_k \sim BETA(\alpha, \beta = t\alpha)$, where $\frac{1}{1+t} = p_{prior}$. This results in: $E(\theta_k) = p_{prior}$.
`dat`	An $n \times s$ matrix of binary outcomes, where *n* is the number of observations (e.g., stream observations over time) and *s* is the number experimental units observed, (e.g., stream segments).
`length`	An optional $n \times 1$ vector containing stream segement lengths to allow calculation of mean stream Bernoulli stream length and mean communication distance.
`w`	Weight for the prior distribution compared to the actual data (generally a proportion).

Details

As our Bayesian framework we assume a conjugate beta prior $\theta_k \sim BETA(\alpha, \beta)$ and binomial likelihood $\boldsymbol{x}_k \mid \theta_k \sim BIN(n, \theta_k)$ resulting in the posterior $\theta_k \mid \boldsymbol{x}_k \sim BETA(\alpha + \sum \boldsymbol{x}_k, \beta + n - \sum \boldsymbol{x}_k)$.

Value

Returns a list with components:

`alpha`	The α shape parameters for the beta and inverse beta posteriors.
`beta`	The β shape parameters for the beta and inverse beta posteriors.
`mean`	The means of the beta posteriors.
`var`	The variances of the beta posteriors.
`mean.inv`	The means of the inverse-beta posteriors.
`var.inv`	The variances of the inverse-beta posteriors.

Com.dist	If `length` is supplied, the mean communication distances of the network.
Length	If `length` is supplied, the mean stream length of the network.
x	The observed number of Bernoulli successes over n trials observed in `dat`.

Author(s)

Ken Aho

See Also

`dinvbeta`.

---

biv.bern *Bivariate Bernoulli Distribution*

Description

Densities (probabilities) of a bivariate Bernoulli distribution, Y_1, Y_2.

Usage

```
biv.bern(p11, p10, p01, p00, y1, y2)
```

Arguments

p11	The probability that $y_1, y_2 = 1, 1$.
p10	The probability that $y_1, y_2 = 1, 0$.
p01	The probability that $y_1, y_2 = 0, 1$.
p00	The probability that $y_1, y_2 = 0, 0$.
y1	Outcome for Y_1.
y2	Outcome for Y_2.

Value

Densities (probability) of the joint Bernoulli distribution.

Author(s)

Ken Aho

Examples

```
biv.bern(0.25,0.25,0.25,0.25,1,0)
biv.bern(0.1,0.4,0.3,0.2,1,0)
```

dc_arc_pres_abs *Stream segment presence absence data for Dry Cr. Idaho*

Description

Stream segment presence absence data for Dry Cr. Idaho (outlet coordinates: 43.71839°N, 116.13747°W). Arc outcomes determined from STIC (Stream Temperature, Intermittency, and Conductivity) sensors at bounding nodes.

Usage

```
data("dc_arc_pres_abs")
```

Format

A data frame with 46187 observations on the following 29 variables.

`datetime` a POSIXlt

'DC10-->C1' a numeric vector

'C1-->DC12' a numeric vector

'DC11-->C1' a numeric vector

'DC12-->C2' a numeric vector

'C2-->DC15' a numeric vector

'DC13-->C2' a numeric vector

'DC15-->C3' a numeric vector

'C3-->DC16' a numeric vector

'DC14-->C3' a numeric vector

'DC16-->C4' a numeric vector

'C4-->DC19' a numeric vector

'DC17-->C5' a numeric vector

'C5-->C4' a numeric vector

'DC18-->C5' a numeric vector

'DC19-->C6' a numeric vector

'C6-->DC4' a numeric vector

'DC20-->C6' a numeric vector

'DC4-->C7' a numeric vector

'C7-->DC5' a numeric vector

'DC1-->DC2' a numeric vector

'DC2-->DC3' a numeric vector

'DC3-->C7' a numeric vector

'DC5-->C8' a numeric vector

'C8-->DC6' a numeric vector

'DC9-->C9' a numeric vector

'C9-->DC7' a numeric vector

'DC8-->C9' a numeric vector

'DC7-->C8' a numeric vector

Source

Maggie Kraft

`dc_lengths` *Lengths of Dry Creek stream (arc) segments*

Description

Lengths of stream (arc) segments from Dry Creek Idaho (outlet coordinates: 43.71839°N, 116.13747°W).

Usage

```
data("dc_lengths")
```

Format

A data frame with 28 observations on the following 2 variables.

`Arcs` Arc names, arrows directionally connect nodes.

`Lengths` Stream segment (arc) length in meters.

Source

Maggie Kraft

`dc_node_pres_abs` *Stream node presence absence data for Dry Cr. Idaho*

Description

Stream node surface water presence absence at Dry Creek ID (outlet coordinates: 43.71839°N, 116.13747°W). Outcomes based on STIC (Stream Temperature, Intermittency, and Conductivity) sensor responses, resulting in binary observations for 29 nodes at 15 minutes intervals, over three years.

Usage

```
data("dc_node_pres_abs")
```

Format

A data frame with 46187 observations on the following 30 variables.

`datetime` a POSIXlt

`DC10` a numeric vector

`C1` a numeric vector

`DC11` a numeric vector

`DC12` a numeric vector

`C2` a numeric vector

`DC13` a numeric vector

`DC15` a numeric vector

`C3` a numeric vector

`DC14` a numeric vector

`DC16` a numeric vector

`C4` a numeric vector

`DC17` a numeric vector

`C5` a numeric vector

`DC18` a numeric vector

`DC19` a numeric vector

`C6` a numeric vector

`DC20` a numeric vector

`DC4` a numeric vector

`C7` a numeric vector

`DC1` a numeric vector

`DC2` a numeric vector

DC3 a numeric vector

DC5 a numeric vector

C8 a numeric vector

DC9 a numeric vector

C9 a numeric vector

DC8 a numeric vector

DC7 a numeric vector

DC6 a numeric vector

Source

Maggie Kraft

---

degree.dists *Potential degree distributions*

Description

Calculates degree distribution probability density. By default calculates an uncorrelated (random) density for a given degree.

Usage

```
degree.dists(d, exp.lambda = 3/2, normalize = TRUE)
```

Arguments

d	degree
exp.lambda	if not NULL, allows specification of chaotic exp.lambda < 3/2 and correlated stochastic processes exp.lambda < 3/2.
normalize	ensures that sum of demsities = 1

Details

In general $f(d) = \exp(-\lambda d)$ where d is the degree. For random degree distributions, $\lambda = \log(3/2)$.

Value

Returns a density plot for a degree.

Author(s)

Ken Aho

See Also

degree.distribution, plot_degree.dist.

`delete.arcs.pa` *Delete arcs based on presence absence data*

Description

Create a new graph after deleting stream graph arcs based on presence/absence data, e.g., data based on outcomes from STIC (Stream Temperature, Intermittency, and Conductivity) loggers.

Usage

```
delete.arcs.pa(G, pa)
```

Arguments

`G`	A graph object of class "igraph", see `graph_from_literal`
`pa`	A vector of binary = 0,1 values indicating the absence or presence of arcs from `E(G)`.

Value

Returns a *igraph* graph object missing the arcs indicated with 0 in `pa`.

Author(s)

Ken Aho, Gabor Csardi wrote `delete.edges`

Examples

```
G <- graph_from_literal(a--+b--+c--+d--+e)
delete.arcs.pa(G, c(0,0,1,1))
```

`delete.nodes.pa` *Delete nodes based on presence absence data*

Description

Create a new graph after deleting stream graph nodes based on presence/absence data, e.g., data based on outcomes from STIC (Stream Temperature, Intermittency, and Conductivity) loggers.

Usage

```
delete.nodes.pa(G, pa, na.response = "none")
```

Arguments

G	A graph object of class "igraph", see `graph_from_literal`
pa	A vector of binary = 0,1 values indicating the absence or presence of nodes from `V(G)`.
na.response	One of `"none"`, `"treat.as.0"`, or `"treat.as.1"`.

Details

A perennial problem with STIC (Stream Temperature, Intermittency, and Conductivity) sensors is the presence of missing data. If `na.response = "none"` and NAs exist then the waring message `"NAs in data need to be addressed. NAs converted 0."` is printed. One can also choose `na.response = "treat.as.0"` or `na.response = "treat.as.1"` which converts NAs to zeroes or ones. Clearly, none of these draconian approaches is optimal. Thus if NAs occur, an attribute is added to the output graph object returned by function which lists the nodes with missing data. This attribute can be obtained with `out$$NA.vertices` where `out <- delete.nodes.pa(...)`, see Examples below. An alternative is to use a classification algorithm for imputation e.g., `missForest::missForest`.

Value

Returns a *igraph* graph object, missing the nodes indicated with 0 in `pa`.

Author(s)

Ken Aho, Gabor Csardi wrote `delete.vertices`

Examples

```
G <- graph_from_literal(a--+b--+c--+d--+e)
delete.nodes.pa(G, c(0,0,1,1,1))
# delete.nodes.pa(G, c(0,0,NA,1,1)) # gives warning and converts NA to 0
d <- delete.nodes.pa(G, c(0,0,NA,1,1), "treat.as.0")
d
d$NA.vertices
```

---

dinvbeta — *Inverse Beta Distribution*

Description

Calculates density (`dinvbeta`), lower-tailed probability (`pinvbeta`) and obtains random outcomes (`rinvbeta`) for an inverse beta distribution

Usage

```
dinvbeta(x, alpha, beta)
pinvbeta(x, alpha, beta)
rinvbeta(n, alpha, beta)
```

Arguments

`x`	Quantile vector or scalar at which to evaluate density or probability.
`alpha`	Alpha parameter
`beta`	Beta parameter
`n`	Number of random outcomes to be generated.

Value

Returns density, probability, and random outcomes for an inverse beta distribution.

Author(s)

Ken Aho and Dwayne Derryberry

See Also

See Also `dbeta`.

Examples

```
dinvbeta(1,1,1)
pinvbeta(1,1,1)
rinvbeta(1,1,1)
```

`efficiency` *Local and global efficiency*

Description

Efficiency is the reciprocal of internodal distance. Thus, the efficiency beween nodes i and j is defined as $e_{i,j} = \frac{1}{d_{i,j}}$ where $d_{i,j}$ denotes the distance between nodes i and j for all $i \neq j$.

Usage

```
efficiency.matrix(G, mode = "in")

avg.efficiency(G, mode = "in")

global.efficiency(G, mode = "in")
```

Arguments

`G`	Graph object of class "igraph". See `graph_from_literal`.
`mode`	One of `"in"` or `"out"`. The former considers in-path efficiencies, whereas the latter considers out-paths.

Details

The function `efficiency.matrix` calculates an efficiency matrix whose elements correspond to elements in the graph distance matrix. The function `avg.efficiency` calculates average efficiencies of nodes to all other nodes, thus providing a local measure of graph connectedness. The function `global.efficiency` calculates the mean of the of all pairwise efficiencies, thus providing a global measure of graph connectedness. For all three functions, reciprocals of infinite distances are taken to be zero.

Value

The function `efficiency.matrix` returns a reciprocal distance matrix for nodes in `G`. The function `avg.efficiency` treats efficiency as a local measure, and thus returns a vector whose entries are average efficiencies for each node. The function `global.efficiency` returns a scalar (the mean of the reciprocal distance matrix).

Author(s)

Ken Aho. Gabor Csardi wrote the function `distances` in *igraph*.

References

Ek, B., VerSchneider, C., & Narayan, D. A. (2015). Global efficiency of graphs. *AKCE International Journal of Graphs and Combinatorics, 12*(1), 1-13.

Examples

```
kon_full <- streamDAGs("konza_full")
efficiency.matrix(kon_full)
avg.efficiency(kon_full)
global.efficiency(kon_full)
```

`gj_coords16` *Coordinates of nodes at Gibson Jack Creek, Idaho for a 2016 survey*

Description

Latitudes and Longitudes of nodes established at Gibson Jack in 2016. Datum: WGS 84.

Usage

```
data("gj_coords16")
```

Format

A data frame with 124 observations on the following 3 variables.

`Object.ID` Node name

`lat` Latitude

`long` Longitude

gj_lengths *Lengths of Gibson Jack stream (arc) segments*

Description

Lengths of stream (arc) segments from the Gibson Jack watershed in southeast Idaho (outlet coordinates: 42.767180°N, 112.480240°W).

Usage

```
data("gj_lengths")
```

Format

A data frame with 28 observations on the following 2 variables.

`Arcs` Arc names, arrows directionally connect nodes.

`Lengths` Stream segment (arc) length in meters.

Source

Maggie Kraft

gj_node_pres_abs *Stream node presence absence data for Gibson Jack Idaho*

Description

Stream node surface water presence absence data from Gibson Jack, Idaho (outlet coordinates: 42.767180°N, 112.480240°W). Outcomes based on STIC (Stream Temperature, Intermittency, and Conductivity) sensors, resulting in binary observations for 29 nodes at 15 minutes intervals over three years.

Usage

```
data("gj_node_pres_abs")
```

Format

A data frame with 55109 observations on the following 30 variables.

`datetime` a POSIXlt

`GJ16` a numeric vector

`GJ15` a numeric vector

`GJ14` a numeric vector

C2 a numeric vector

C3 a numeric vector

C4 a numeric vector

GJ11 a numeric vector

GJ13 a numeric vector

GJ12 a numeric vector

GJ20 a numeric vector

GJ18 a numeric vector

GJ17 a numeric vector

C5 a numeric vector

GJ10 a numeric vector

GJ9 a numeric vector

C6 a numeric vector

GJ23 a numeric vector

GJ22 a numeric vector

C1 a numeric vector

C7 a numeric vector

GJ24 a numeric vector

GJ21 a numeric vector

GJ8 a numeric vector

GJ7 a numeric vector

GJ3 a numeric vector

C8 a numeric vector

GJ6 a numeric vector

GJ5 a numeric vector

GJ4 a numeric vector

Source

Maggie Kraft

gj_node_pres_abs16 *Stream node presence absence data for Gibson Jack Cr. Idaho, for a 2016 survey*

Description

Streamflow presence and absence data for each node location collected by manual observation November 6, 2016, May 6, 2017, and August 14, 2017. Note, a longer dataset 2021-2023, gathered by the AIMS team at fewer points, is available for Gibson Jack under the name `gj_node_pres_abs`.

Usage

```
data("gj_node_pres_abs16")
```

Format

A data frame with 3 observations on the following 125 variables.

`Date` a character vector

`GJ_ST1_0600` a numeric vector

`GJ_ST1_0400` a numeric vector

`GJ_ST1_0200` a numeric vector

`GJ_ST1_0000` a numeric vector

`GJ_SF_2800` a numeric vector

`GJ_SF_2600` a numeric vector

`GJ_SF_2400` a numeric vector

`GJ_SF_2200` a numeric vector

`GJ_SF_2000` a numeric vector

`GJ_SF_1800` a numeric vector

`GJ_SF_1600` a numeric vector

`GJ_SF_1400` a numeric vector

`GJ_SF_1200` a numeric vector

`GJ_SF_1000` a numeric vector

`GJ_SF_0800` a numeric vector

`GJ_SF_0600` a numeric vector

`GJ_SF_0400` a numeric vector

`GJ_SF_0200` a numeric vector

`GJ_SF_0000` a numeric vector

`GJ_NT1_WF_FH` a numeric vector

`GJ_NT1_WF_000` a numeric vector

GJ_NT1_0800 a numeric vector

GJ_NT1_0600 a numeric vector

GJ_NT1_0400 a numeric vector

GJ_NT1_0200 a numeric vector

GJ_NT1_0000 a numeric vector

GJ_NT2_1600 a numeric vector

GJ_NT2_1400 a numeric vector

GJ_NT2_1200 a numeric vector

GJ_NT2_1000 a numeric vector

GJ_NT2_0800 a numeric vector

GJ_NT2_0600 a numeric vector

GJ_NT2_0400 a numeric vector

GJ_NT2_0200 a numeric vector

GJ_NT2_0000 a numeric vector

GJ_NT3_0200 a numeric vector

GJ_NT3_0000 a numeric vector

GJ_NT4_0600 a numeric vector

GJ_NT4_0400 a numeric vector

GJ_NT4_0200 a numeric vector

GJ_NT4_0000 a numeric vector

GJ_NF_3800 a numeric vector

GJ_NF_3750 a numeric vector

GJ_NF_3600 a numeric vector

GJ_NF_3400 a numeric vector

GJ_NF_3200 a numeric vector

GJ_NF_3000_CU a numeric vector

GJ_NF_3000_CD a numeric vector

GJ_NF_2800_CU a numeric vector

GJ_NF_2800_CD a numeric vector

GJ_NF_2600 a numeric vector

GJ_NF_2400_CU a numeric vector

GJ_NF_2400_CD a numeric vector

GJ_NF_2200 a numeric vector

GJ_NF_2000 a numeric vector

GJ_NF_1800 a numeric vector

GJ_NF_1600 a numeric vector

GJ_NF_1400 a numeric vector

GJ_NF_1200 a numeric vector

GJ_NF_1060_CU a numeric vector

GJ_NF_1060_CD a numeric vector

GJ_NF_1000 a numeric vector

GJ_NF_0800 a numeric vector

GJ_NF_0600 a numeric vector

GJ_NF_0400 a numeric vector

GJ_NF_0200 a numeric vector

GJ_NF_0000 a numeric vector

GJ_MT2_0900 a numeric vector

GJ_MT2_0800 a numeric vector

GJ_MT2_0600 a numeric vector

GJ_MT2_0400 a numeric vector

GJ_MT2_0200 a numeric vector

GJ_MT2_0000 a numeric vector

GJ_MT1_1650 a numeric vector

GJ_MT1_1600 a numeric vector

GJ_MT1_1550 a numeric vector

GJ_MT1_1500 a numeric vector

GJ_MT1_1450 a numeric vector

GJ_MT1_1400 a numeric vector

GJ_MT1_1350 a numeric vector

GJ_MT1_1300 a numeric vector

GJ_MT1_1250 a numeric vector

GJ_MT1_1200 a numeric vector

GJ_MT1_1150 a numeric vector

GJ_MT1_1100 a numeric vector

GJ_MT1_1050 a numeric vector

GJ_MT1_1000 a numeric vector

GJ_MT1_0950 a numeric vector

GJ_MT1_0900 a numeric vector

GJ_MT1_0850 a numeric vector

GJ_MT1_0800 a numeric vector

GJ_MT1_0750 a numeric vector

GJ_MT1_0700 a numeric vector

GJ_MT1_0650 a numeric vector

GJ_MT1_0600 a numeric vector

GJ_MT1_0550 a numeric vector

GJ_MT1_0500 a numeric vector

GJ_MT1_0450 a numeric vector

GJ_MT1_0400 a numeric vector

GJ_MT1_0350 a numeric vector

GJ_MT1_0300 a numeric vector

GJ_MT1_0250 a numeric vector

GJ_MT1_0200 a numeric vector

GJ_MT1_0150 a numeric vector

GJ_MT1_0100 a numeric vector

GJ_MT1_0050 a numeric vector

GJ_MT1_0000 a numeric vector

GJ_MS_3000 a numeric vector

GJ_MS_2800 a numeric vector

GJ_MS_2600 a numeric vector

GJ_MS_2400 a numeric vector

GJ_MS_2200 a numeric vector

GJ_MS_2000 a numeric vector

GJ_MS_1800_CU a numeric vector

GJ_MS_1800_CD a numeric vector

GJ_MS_1600 a numeric vector

GJ_MS_1400 a numeric vector

GJ_MS_1200 a numeric vector

GJ_MS_1000 a numeric vector

GJ_MS_0800 a numeric vector

GJ_MS_0600 a numeric vector

GJ_MS_0400 a numeric vector

GJ_MS_0200 a numeric vector

GJ_MS_0000 a numeric vector

global.summary *Global Summary*

Description

This function calculates useful DAG global summaries including size, diameter, number of paths to sink, mean path length, mean alpha centrality, mean PageRank centrality, graph centralization, Strahler order, Shreve order, the Randic index, the first Zagreb Index, the second Zagreb index, atom-bond connectivity, the geometric-arithmatic index, the harmonic index, the Harary index, global efficiency, the assortativity correlation (+, -), and the assortativity correlation (+, +).

Usage

```
global.summary(G, which = "all", sink, mode = "in", inf.paths = FALSE)
```

Arguments

`G`	graph object of class "igraph". See `graph_from_literal`.
`which`	Which metric to use. Currently one of `"all"`, `"size"`, `"diameter"`, `"graph.order"`, `"n.sources"`, `"n.paths.to.sink"`, `"sink.path.len.summary"`, `"deg.summary"`, `"avg.alpha.cent"`, `"shreve.num"`, `"strahler.num"`, `"fst.zagreb"`, `"scd.zagreb"`, `"ABC"`, `"harary"`, `"global.efficiency"`, `"assort.in.out"`, `"assort.in.in"`.
`sink`	sink node from graph object `G`.
`mode`	Type of degree used. One of `"in"` or `"out"`.
`inf.paths`	logical, consider infinite paths?

Details

Simple global graph measures of complexity and/or connectivity of a stream DAG include *size*, *diameter*, and number of paths to a sink. The *size* is equal to the number of arcs in the stream network. The *diameter* equals the length of the longest path, i.e., the *height* of the sink, and *in eccentricity* of the sink. The number of paths to the sink is equivalent to the number of nodes from which the sink node is reachable, which will be $n-1$ for a fully active stream. For more information on $I(D)$ metrics see `I.D`. Links describing other metrics are provided below.

Value

Returns a vector of global graph measures for `G`.

Author(s)

Ken Aho, Gabor Csardi wrote `alpha_centrality` and other underlying functions.

References

Kunkler, S. J., LaMar, M. D., Kincaid, R. K., & Phillips, D. (2013). Algorithm and complexity for a network assortativity measure. arXiv Preprint *arXiv:1307.0905*.

Das, K. C., Gutman, I., & Furtula, B. (2011). On atom-bond connectivity index. *Chemical Physics Letters, 511*(4-6), 452-454.

Li, X., & Shi, Y. (2008). A survey on the randic index. *MATCH Commun. Math. Comput. Chem, 59*(1), 127-156.

See Also

alpha_centrality, I.D, spath.lengths, n.sources, stream.order, harary

Examples

```
network_a <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e,
e --+ j, j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l,
l --+ m, m --+ n,  n --+ o)

global.summary(network_a, sink ="o")
```

---

harary *Harary Index*

Description

Computes the Harary global metric for a stream DAG.

Usage

```
harary(G)
```

Arguments

G	Graph object of class igraph. See graph_from_literal.

Details

The Harary index is computed as:

$$\frac{1}{2}\sum_{i}^{m}\sum_{j}^{m}(RD)_{ij}$$

where $(RD)_{ij}$ is the reciprocal of the ijth element of the graph distance matrix. Reciprocals of infinite values in the distance matrix are taken to be zero. Users should be aware that the graph object G is assumed to be DAG, and that distances are based on in-paths.

Value

Returns a scalar: the global Harary index.

Author(s)

Ken Aho, Gabor Csardi wote distances

References

Plavsic, D., Nikolic, S., Trinajstic, N., & Mihalic, Z. (1993). On the Harary index for the characterization of chemical graphs. Journal of Mathematical Chemistry, 12(1), 235-250.

Examples

```
harary(streamDAGs("konza_full"))
```

I.D *Generalized DAG indices*

Description

Calculates global generalized topopological indices for a digraph

Usage

```
I.D(G, mode = "gen.rand", alpha = -1/2, mult = FALSE, degrees = "out.in")
```

Arguments

G	Graph object of class. See graph_from_literal.
mode	One of "gen.rand","gen.sum.con", "ABC", "GA", "harm", "aug.rand".
alpha	Exponent value for forms of omega with alpha exponent.
mult	Logical if TRUE use experimental multiplicative measures.
degrees	Degree designations for the arc $\vec{uv}$. One of "out.in", "out.out", "in.out", "in.in". See Details below. The default designation, "out.in", is strongly reccomended for stream DAGs.

Details

For an arc $a = \overrightarrow{uv}$, $a \in A$, we denote the out degree of u as d_u^+, and the in degree of v as d_v^-. Now let $I(D)$ represent a generalized topopological index for a digraph, D (cf. Deng et al., 2021) that depends on d_u^+ and d_v^-:

$$I(D) = 1/2 \sum_{uv \in A} \omega(d_u^+, d_v^-)$$

Six basic configurations for $I(D)$ can be recognized:

1. If $\omega(x, y) = (xy)^\alpha$, for $\alpha \neq 0$, then $I(D)$ is the *general directed Randic index* (Kincaid et al., 2016) for D. Specific variants include the *Randic index* ($\alpha = -1/2$), the *second Zagreb index* ($\alpha = 1$) and the *second modified Zagreb index* ($\alpha = -1$) (Anthony & Marr, 2021).

2. If $\omega(x, y) = (x+y)^\alpha$, then $I(D)$ is the *general sum-connectivity index* for *D* (Deng et al., 2021). Further, if $\omega(x, y) = 2(x + y)^\alpha$, then $I(D)$ is the *sum connectivity* (Zhou & Trinajstic, 2009), and the directed *first Zagreb index* (Anthony & Marr, 2021) for $\alpha = -1/2$ and $\alpha = 1$, respectively .

3. If $\omega(x, y) = \sqrt{((x + y - 2)/xy)}$, then $I(D)$ is the *atom bond connectivity* of *D* (Estrada et al., 1998).

4. If $\omega(x, y) = \sqrt{xy}/(1/2(x + y))$, then $I(D)$ is the *geometric-arithmetic index* for D (Vukicevic & Furtula, 2009).

5. If $\omega(x, y) = 2/(x + y)$, then $I(D)$ is the *harmonic index* of D (Favaron et al., 1993).

6. If $\omega(x, y) = \left(\frac{xy}{x+y-2}\right)^3$, then $I(D)$ is the *augmented Randic index* of D (Furtula et al. 2010). This index is not reccomended for stream DAGs as it will contained undefined terms for any network with unbranched paths.

More options are possible under the generalization of Kincaid (1996). Specifically, for an arc $a = \vec{uv}, a \in A$, let $\gamma, \tau \in -, +$ index the degree type: $- = in, + = out$. Then, four combinations of d_u^γ, d_v^τ can occur, resulting in four different versions of each $I(D)$ metric described above. These combinations are: d_u^+, d_v^- (as shown above), d_u^+, d_v^+, d_u^-, d_v^-, and d_u^-, d_v^+. The default d_u^+, d_v^- is strongly reccomended for stream DAGs over other variants.

Value

Index values for a DAG

Author(s)

Ken Aho, Gabor Csardi wrote `degree`

References

Anthony, B. M., & Marr, A. M. (2021). Directed zagreb indices. *Graphs and Combinatorial Optimization: From Theory to Applications: CTW 2020 Proceedings*, 181-193.

Deng, H., Yang, J., Tang, Z., Yang, J., & You, M. (2021). On the vertex-degree based invariants of digraphs. arXiv Preprint *arXiv:2104.14742*.

Estrada, E., Torres, L., Rodriguez, L., & Gutman, I. (1998). *An atom-bond connectivity index: Modelling the enthalpy of formation of alkanes*. NISCAIR-CSIR, India.

Favaron, O., Maheo, M., & Sacle, J.-F. (1993). Some eigenvalue properties in graphs (conjectures of graffitii). *Discrete Mathematics, 111*(1-3), 197-220.

Furtula, B., Graovac, A., & Vukicevic, D. (2010). Augmented Zagreb index. *Journal of Mathematical Chemistry, 48*(2), 370-380.

Kincaid, R. K., Kunkler, S. J., Lamar, M. D., & Phillips, D. J. (2016). Algorithms and complexity results for finding graphs with extremal Randic index. Networks, 67(4), 338-347.

Vukicevic, D., & Furtula, B. (2009). Topological index based on the ratios of geometrical and arithmetical means of end-vertex degrees of edges. *Journal of Mathematical Chemistry, 46*(4), 1369-1376.

Zhou, B., & Trinajstic, N. (2009). On a novel connectivity index. *Journal of Mathematical Chemistry*, 46(4), 1252-1270.

See Also

degree

Examples

```
network_a <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e,
e --+ j, j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l,
l --+ m, m --+ n,  n --+ o)
I.D(network_a)
```

ICSL *Integral connectivity scale length (ICSL)*

Description

Integral connectivity scale lengths (ICSL, Western et al. 2013) is the average distance between wet locations using either (1) Euclidean distance or (2) topographically-defined hydrologic distance, e.g., instream hydrologic distance, subsurface distance (Ali and Roy 2009) and outlet distance, in which connected saturated paths must reach the catchment outlet.

Usage

```
ICSL(G, coords = NULL, names = NULL, lengths = NULL,
dist.matrix = NULL, show.dist = FALSE)
```

Arguments

G	A graph object of class "igraph", see graph_from_literal
coords	Spatial coordinates to allow computation of nodal Euclidean distances
names	Nodal names
lengths	Stream arc lengths or hydrologic arc lengths
show.dist	Logical. Show distance matrix?
dist.matrix	An optional distance matrix, potentially providing non-Euclidean node distances (e.g., node subsurface distance, etc.). Distance matrix Labels in dist.matrix must be analogous to those used in G. Note that dimensions in dist.matrix can be larger than the number of nodes in G if, for instance, dist.matrix represents distances of the complete wetted network and G is a subgraph of the complete wetted network after drying.

Details

Computes either: 1) the average Euclidean distance of connected nodal locations as defined in `G`, if `coords` are provided, 2) if `dist.matrix` is provided, the average nodal distances of a distance matrix provided in `dist.matrix` for nodes that remain in `G`, or 3) the instream distances of connected nodal locations if stream lenghts are provided in `lengths`. For 3), the `length` vector will need to be trimmed as arcs disappear within intermittent streams (see Examples).

Value

Returns a global distance scalar. See **Details**.

Author(s)

Ken Aho, Gabor Csardi wrote underlying functions `distances` and `E`

References

Ali, G. A., & Roy, A. G. (2010). Shopping for hydrologically representative connectivity metrics in a humid temperate forested catchment. *Water Resources Research, 46*(12).

Western, A. W., Bloschl, G., & Grayson, R. B. (2001). Toward capturing hydrologically significant connectivity in spatial patterns. *Water Resources Research, 37*(1), 83-97.

See Also

`distances`

Examples

```
murphy_spring <- graph_from_literal(IN_N --+ M1984 --+ M1909, IN_S --+ M1993,
M1993 --+ M1951 --+ M1909 --+ M1799 --+ M1719 --+ M1653 --+ M1572 --+ M1452,
M1452 --+ M1377 --+ M1254 --+ M1166 --+ M1121 --+ M1036 --+ M918 --+ M823,
M823 --+ M759 --+ M716 --+ M624 --+ M523 --+ M454 --+ M380 --+ M233 --+ M153,
M153 --+ M91 --+ OUT)

#---- ICSL based on nodal Euclidean distances ----#
data(mur_coords)
ICSL(murphy_spring, coords = mur_coords[,2:3], names = mur_coords[,1])

#---- ICSL based on in-stream length data ----#
data(mur_lengths)
ICSL(murphy_spring, lengths = mur_lengths[,2], names = mur_coords[,1])

# or, simply
ms <- murphy_spring
E(ms)$weight <- mur_lengths[,2]
ICSL(ms)

# Arcs 1 and 3 dry
B <- graph_from_literal(IN_N, M1984, IN_S --+ M1993 --+ M1951 --+ M1909,
M1909 --+ M1799 --+ M1719 --+ M1653 --+ M1572 --+ M1452 --+ M1377 --+ M1254,
M1254 --+ M1166 --+ M1121 --+ M1036 --+ M918 --+ M823 --+ M759 --+ M716,
```

```
M716 --+ M624 --+ M523 --+ M454 --+ M380 --+ M233 --+ M153 --+ M91 --+ OUT)
ICSL(B, lengths = mur_lengths[,2][-c(1,3)], show.dist = TRUE)
```

`imp.closeness` *Improved Closeness Centrality*

Description

Calculates improved closeness centrality of individual nodes in a DAG.

Usage

```
imp.closeness(G)
```

Arguments

`G`	Graph object of class "igraph", see See `graph_from_literal`.

Details

Improved closeness centrality (Beauchamp, 1965) was developed for weakly connected or disconnected digraphs. The measure is based on the reciprocal of nodal shortest path distances from the *j*th node to the *k*th node, $1/\delta_{j,k}$. For the *j*th node this is:

$$H_j = (n-1)\sum_{j\neq k} 1/\delta_{j,k}$$

where, for disconnected nodes, the reciprocal distance $1/\infty$ is taken to be zero.

Value

Improved closeness centrality of a node

Author(s)

Ken Aho, Gabor Csardi wrote `distances`

References

Beauchamp, M. A. (1965). An improved index of centrality. *Behavioral Science, 10*(2), 161-163.

See Also

`distances`

Examples

```
network_a <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e,
e --+ j, j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l,
l --+ m, m --+ n,  n --+ o)
imp.closeness(network_a)
```

`isle` *Detects and defines islands in a streamDAG*

Description

The function was written primarilly to recognize DAG islands to allow correct implementation of the function `stream.order` and is still early in its development.

Usage

```
isle(G)
```

Arguments

G	Graph object of class `igraph`. See `graph_from_literal`.

Details

The function currently allows detection of simple island structures (those that don't contain sub-islands). One of the output objects from the function is a new graph object with island nodes into a single node(s).

Value

Output consists of the following:

test	Logical indicating whether or not `G` contains islands.
island	List of islands with their nodal components
input.id	Neighboring node(s) directly upstream from island(s).
output.id	Neighboring node(s) directly downstream from island(s).
new.graph	New graph object created from `G` in which nodes constituing islands a combined into a single node
island.names	Names of island nodes created in new output graph (that combines nodes constituing islands into a single node). Follows the naming system `"i-1"`, `"i-2"`, etc.
splits	The number of islands detected.

Author(s)

Ken Aho

See Also

stream.order, delete.vertices, codeadd.vertices, codeadd.edges

Examples

```
G <- graph_from_literal(a --+ c --+ e, b --+ d --+ e --+ f --+ p, g --+ i --+ j --+ m,
i --+ k --+ m, m --+ n --+ o --+ p, h --+ l --+ n, p --+ q --+ r)
plot(G)
isle(G)
```

---

jd_lengths *Lengths of Johnson Draw stream (arc) segments*

Description

Lengths of stream (arc) segments from Johnson Draw in southwest Idaho (outlet coordinates: 43.12256°N, 116.77630°W). The dataframe `jd_lengths` contains segment lengths in the absence of piezos [nodes are currently defined by STIC (Stream Temperature, Intermittency, and Conductivity) locations only] and thus correspond to the network in `streamDAGs("jd_full")`. A network that includes piezos is depicted by `streamDAGs("jd_piezo_full")`.

Usage

```
data("jd_lengths")
```

Format

A data frame with observations on the following 2 variables.

`Arcs` Arc names, arrows directionally connect nodes.

`Lengths` Lengths in meters

Source

Arya Legg, Maggie Kraft

jd_node_pres_abs *Stream node presence absence data for Johnson Draw Idaho*

Description

Stream node surface water presence absence data for the Johnson Draw watershed in southwest Idaho (outlet coordinates: 43.12256°N, 116.77630°W). Outcomes based on STC sensors, resulting in binary observations for 22 nodes at 15 minutes intervals over three years.

Usage

```
data("jd_node_pres_abs")
```

Format

A data frame with 50322 observations on the following 23 variables.

`datetime` a POSIXlt

`JD5` a numeric vector

`JD6` a numeric vector

`JD7` a numeric vector

`C1` a numeric vector

`JD10` a numeric vector

`JD9` a numeric vector

`JD8` a numeric vector

`JD11` a numeric vector

`JD12` a numeric vector

`JD13` a numeric vector

`C2` a numeric vector

`JD16` a numeric vector

`JD17` a numeric vector

`JD18` a numeric vector

`JD19` a numeric vector

`JD20` a numeric vector

`JD4` a numeric vector

`JD3` a numeric vector

`JD2` a numeric vector

`JD1` a numeric vector

`JD15` a numeric vector

`JD14` a numeric vector

Source

Maggie Kraft

kon_coords *Coordinates of nodes in the Konza Praire dataset*

Description

Coordinates (in Lat/Long) of nodes established at the Konza Prairie stream network.

Usage

```
data("kon_coords")
```

Format

A data frame with 46 observations on the following 3 variables.

`Object.ID` Node name

`lat` Latitude

`long` Longitude

kon_lengths *Lengths of Murphy Cr. stream (arc) segments*

Description

Lengths of Murphy Cr. stream (arc) segments

Usage

```
data("kon_lengths")
```

Format

A data frame with 45 observations on the following 2 variables.

`Arcs` Arc names, arrows directionally connect nodes.

`Lengths` Lengths in meters

Source

Rob Ramos

`local.summary` *local (nodal) summaries of a DAG*

Description

Obtains local (nodal) summaries from a DAG

Usage

```
local.summary(G, metric = "all", mode = "in")
```

Arguments

`G`	Graph of class "igraph". See `graph_from_literal`
`metric`	One of `"all"`, `"alpha.cent"`, `"imp.closeness"`, `"n.paths"`, `"n.nodes"`, `"page.rank"`, `"path.len.summary"`, `"path.deg.summary"`, `"size.intact.in"`. Partial matches allowed.
`mode`	One of `"in"` or `"out"`

Value

Nodes are returned with values measuring the indegree, alpha centrality, PageRank centrality, improved closeness centrality, betweenness centrality, upstream network length, and upstream in-path length mean, variance, max (i.e., in-eccentricity), skew, kurtosis, and mean efficiency.

Author(s)

Ken Aho, Gabor Csardi wrote `degree`, `page_rank` and `alpha_centrality` functions.

See Also

`degree`, `alpha_centrality`, `page_rank`, `betweenness`, `imp.closeness`, `skew`, `kurt`

Examples

```
network_a <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e,
e --+ j, j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l,
l --+ m, m --+ n,  n --+ o)
local.summary(network_a)
```

mur_arc_pres_abs *Stream segment presence absence data for Murphy Cr. Idaho*

Description

Simulated multivariate Benroulli outcomes for 27 stream segments, based on their observed marginal probabilities for steam presence and covariance structures. "M"-labelling for nodes indicates "meters above outlet".

Usage

```
data("mur_arc_pres_abs")
```

Format

A data frame with 1000 observations on the following 27 variables.

'IN_N-->M1984' a numeric vector

'M1984-->M1909' a numeric vector

'M1909-->M1799' a numeric vector

'IN_S-->M1993' a numeric vector

'M1993-->M1951' a numeric vector

'M1951-->M1909' a numeric vector

'M1799-->M1719' a numeric vector

'M1719-->M1653' a numeric vector

'M1653-->M1572' a numeric vector

'M1572-->M1452' a numeric vector

'M1452-->M1377' a numeric vector

'M1377-->M1254' a numeric vector

'M1254-->M1166' a numeric vector

'M1166-->M1121' a numeric vector

'M1121-->M1036' a numeric vector

'M1036-->M918' a numeric vector

'M918-->M823' a numeric vector

'M823-->M759' a numeric vector

'M759-->M716' a numeric vector

'M716-->M624' a numeric vector

'M624-->M523' a numeric vector

'M523-->M454' a numeric vector

'M454-->M380' a numeric vector

`M380-->M233` a numeric vector

`M233-->M153` a numeric vector

`M153-->M91` a numeric vector

`M91-->OUT` a numeric vector

`mur_coords` *Coordinates of nodes at Murphy Ck. Idaho*

Description

UTM coordinates (Zone 11T) and Latitudes and Longitudes of nodes established at Murphy Cr. Idaho. Datum: WGS 84.

Usage

```
data("mur_coords")
```

Format

A data frame with 28 observations on the following 5 variables.

`Object.ID` Node name

`E` UTM Easting

`N` UTM Northing

`lat` Latitude

`long` Longitude

`mur_lengths` *Lengths of Murphy Cr. stream (arc) segments*

Description

Lengths of Murphy Cr. stream (arc) segments

Usage

```
data("mur_lengths")
```

Format

A data frame with 27 observations on the following 2 variables.

`Arcs` Arc names, arrows directionally connect nodes.

`Lengths` Stream segment (arc) length in meters.

Source

Warix, S. R., Godsey, S. E., Lohse, K. A., & Hale, R. L. (2021). Influence of groundwater and topography on stream drying in semi-arid headwater streams. Hydrological Processes, 35(5), e14185.

`mur_node_pres_abs` *Stream node presence absence data for Murphy Cr. Idaho*

Description

A subset of stream node presence absence data from Warix et al. (2019) resulting in binary observations for 28 nodes at 2.5 hr intervals.

Usage

```
data("mur_node_pres_abs")
```

Format

A data frame with 1163 observations on the following 29 variables.

`Datetime` a character vector

`IN_N` a numeric vector

`M1984` a numeric vector

`M1909` a numeric vector

`IN_S` a numeric vector

`M1993` a numeric vector

`M1951` a numeric vector

`M1799` a numeric vector

`M1719` a numeric vector

`M1653` a numeric vector

`M1572` a numeric vector

`M1452` a numeric vector

`M1377` a numeric vector

`M1254` a numeric vector

`M1166` a numeric vector

`M1121` a numeric vector

`M1036` a numeric vector

`M918` a numeric vector

`M823` a numeric vector

`M759` a numeric vector

`M716` a numeric vector

`M624` a numeric vector

`M523` a numeric vector

`M454` a numeric vector

`M380` a numeric vector

`M233` a numeric vector

`M153` a numeric vector

`M91` a numeric vector

`OUT` a numeric vector

References

Warix, S. R., Godsey, S. E., Lohse, K. A., & Hale, R. L. (2021). Influence of groundwater and topography on stream drying in semi-arid headwater streams. *Hydrological Processes, 35*(5), e14185.

`mur_seasons_arc_pa` *Simulated seasonal arc presence absence data for Murphy Cr*

Description

A data frame with one hundred multivariate Bernoulli simulated outcomes representing spring, summer and fall.

Usage

```
data("mur_seasons_arc_pa")
```

Format

A data frame with 300 observations on the following 28 variables.

'`IN_N -> M1984`' a numeric vector

'`M1984 -> M1909`' a numeric vector

'`M1909 -> M1799`' a numeric vector

'`IN_S -> M1993`' a numeric vector

'`M1993 -> M1951`' a numeric vector

'`M1951 -> M1909`' a numeric vector

'`M1799 -> M1719`' a numeric vector

'`M1719 -> M1653`' a numeric vector

'`M1653 -> M1572`' a numeric vector

'`M1572 -> M1452`' a numeric vector

'`M1452 -> M1377`' a numeric vector

'`M1377 -> M1254`' a numeric vector

‘M1254 -> M1166’ a numeric vector

‘M1166 -> M1121’ a numeric vector

‘M1121 -> M1036’ a numeric vector

‘M1036 -> M918’ a numeric vector

‘M918 -> M823’ a numeric vector

‘M823 -> M759’ a numeric vector

‘M759 -> M716’ a numeric vector

‘M716 -> M624’ a numeric vector

‘M624 -> M523’ a numeric vector

‘M523 -> M454’ a numeric vector

‘M454 -> M380’ a numeric vector

‘M380 -> M233’ a numeric vector

‘M233 -> M153’ a numeric vector

‘M153 -> M91’ a numeric vector

‘M91 -> OUT’ a numeric vector

Season A categorical variable with three levels: "spring" (6/3/2019 - 7/13/2019), "summer" (7/13/2019 - 8/23/2019) and "fall" (8/23/2019 - 10/2/2019)

---

n.sources *Identify source and sink nodes*

Description

Identify the number of sources and the source nodes. Sources are assumed to be linked to the sink.

Usage

```
n.sources(G, sink = NULL)
sources(G, sink = NULL)
```

Arguments

G	A graph object of class "igraph", see graph_from_literal
sink	The name of the sink node.

Value

Returns a character vector listing streamDAG source nodes (those linked to the sink with indegree 0).

Author(s)

Ken Aho, Gabor Csardi wrote degree

Examples

```
sources(streamDAGs("konza_full"), sink = "SFM01_1")
```

---

path.lengths.sink *Path Lengths*

Description

Obtains all shortest in paths to a sink

Usage

```
path.lengths.sink(G, sink = NULL, inf.paths = TRUE)
```

Arguments

G	Graph object of class "igraph", see: See graph_from_literal.
sink	sink node from G.
inf.paths	Logical, consider infinite paths?

Value

Length of path to a sink

Author(s)

Ken Aho, Gabor Csardi wrote distances

Examples

```
murphy_spring <- graph_from_literal(IN_N --+ M1984 --+ M1909, IN_S --+ M1993,
M1993 --+ M1951 --+ M1909 --+ M1799 --+ M1719 --+ M1653 --+ M1572 --+ M1452,
M1452--+ M1377 --+ M1254 --+ M1166 --+ M1121 --+ M1036 --+ M918 --+ M823,
M823 --+ M759 --+ M716 --+ M624 --+ M523 --+ M454 --+ M380 --+ M233 --+ M153,
M153 --+ M91 --+ OUT)

path.lengths.sink(murphy_spring, sink = "OUT")

# with stream lengths as weights
data(mur_lengths)

E(murphy_spring)$weights <- mur_lengths[,2]
path.lengths.sink(murphy_spring, "OUT")
```

`path.visibility` *Path Visibilities*

Description

Functions detect and summarize visibilities of path nodes from one or several source nodes to an sink. Specifically, the function The function `path.visibility` determines path visibilities from single source node to a single sink. `multi.path.visibility` Generates tables of path visibilities and visibility summaries for multiple source nodes to a single sink.

Ordering of nodes, vitally important to the calculation of visibility is currently obtained by identifying paths from each source node to the sink. The sum of node distances in each path are then sorted decreasingly to define an initial order for calculating visibilities. It is currently assumed that the user will manually handle disconnected paths via the `source` argument of visibility functions. Use of source nodes disconnected to the sink will result in the message: `"only use source nodes connected to sink"`. Because of this situation disconnected graphs will be handled by a function in development `single.node.visibility`.

Usage

```
path.visibility(G, degree = "in", source = NULL, sink = NULL, weights = NULL)

multi.path.visibility(G, degree = "in", source = NULL, sink = NULL,
weights = NULL, autoprint = TRUE)
```

Arguments

`G`	Graph of class "igraph". See `graph_from_literal`
`degree`	One of `"out"` for outdegree, `"in"` for indegree or `"all"` for the sum of the two.
`source`	A starting node for a path. The function `multi.path.visibility` allows multiple starting nodes.
`sink`	An ending node for a path.
`weights`	If `!null`, refers to a $1 \times n$ `data.frame` of weights, with the `data.frame` `name` attribute in `weights` corresponding to node names in `G`.
`autoprint`	Logical. Should table summary of nodal visibilities be automatically printed or made ?

Details

Following Lacasa et al. (2008), let t_a represent the occurrance number of the ath node in a time series or stream path, and let y_a represent a data outcome from the ath node. Nodes a and b will have visibility if all other data, y_c, between a and b fufill:

$$y_c < y_b + (y_a - y_b)\frac{t_b - t_c}{t_b - t_a}.$$

Value

The function `path.visibility` returns a symmetric matrix whose upper triangle denotes nodal co-visibilities. The lower triangle is left empty for efficiency. Reading down a column in the upper triangle shows upstream visibilites to and from a node, while reading across rows shows downstream visibilities.

The function `multi.path.visibility` returns a list containing the three objects. The first is printed and the latter two are `invisible` by default.

`visibility.summary`
: The printed result is a matrix of path visibity counts for a node, with respect to upstream (to), downstream (from), and combined directions (both).

`complete.matrix`
: Analogous, to `path.visibility`, this result attempts to synthesize visibilities within source-to-sink paths for all requested sources into a single matrix.

`all.matrices`
: A list containing `path.visibility` summary matrices for each source-to-sink path.

Output is summarized based on a deduced ordering of nodes from sources to sin. The ordering is based on nodal path lengths.

Author(s)

Ken Aho, Gabor Csardi wrote `degree` and `shortest_paths`.

References

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., & Nuno, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13), 4972-4975.

See Also

`degree`, `shortest_paths`

Examples

```
A <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e,
e --+ j, j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l,
l --+ m, m --+ n,  n --+ o)

path.visibility(A, source = "a", sink = "o")

multi.path.visibility(A, source = c("a","c","f","h"),
sink = "o")

# From Lacasa et al. (2008)

B <- graph_from_literal(a --+ b --+ c --+ d --+ e --+ f --+ g)
weights <- data.frame(matrix(nrow = 1, data = c(0.87, 0.49, 0.36, 0.83, 0.87, 0.49, 0.36)))
```

```
names(weights) = letters[1:7]
path.visibility(B, source = "a", sink = "g", weights = weights)
```

---

plot_degree.dist *Plot degree distributions*

Description

Plots bserved degree distribution against models for uncorrelated random, chaotic and correlated random processes.

Usage

```
plot_degree.dist(G, mode = "all", exp.lambda = c(1.1, 3/2, 2), leg.loc = "topright")
```

Arguments

`G`	Graph object of class "igraph". See `graph_from_literal`
`mode`	Character string, one of `"out"` for out-degree, `"in"` for in-degree or `"all"` for the sum of the two. For undirected graphs this argument is ignored.
`exp.lambda`	log.lamda = if not NULL, allows specification of chaotic `exp.lambda` < `3/2` and correlated stochastic processes `exp.lambda` < `3/2`
`leg.loc`	placement of `legend`,

Value

Plots processes for observed versus distributions under random or chaotic degrees.

Author(s)

Ken Aho

See Also

`degree.dists`, `degree.distribution`

Examples

```
network_a <- graph_from_literal(a --+ b, c --+ d, d --+ e, b --+ e,
e --+ j, j --+ m, f --+ g, g --+ i, h --+ i, i --+ k, k --+ l,
l --+ m, m --+ n,  n --+ o)
plot_degree.dist(network_a)
```

R.bounds *Bounds for the correlation of two (or more) Benrnoulli random variables*

Description

Replaces impossible correlations (values too small or too large) with minimum and maximum correlations, respectively.

Usage

```
min_r(p1, p2)
max_r(p1, p2)
R.bounds(p, R, pad = 0.001)
```

Arguments

p1	Probability of success for first random variable
p2	Probability of success for second random variable
p	Vector of marginal probabilities for multivariate Bernoulli random variables, for R.bounds.
R	Raw correlation matrix for random variables
pad	Padding (in correlation units) to adjust the returned correlation matrix with respect extremal values.

Details

The functions `r.min` and `r.max` define minimum and maximimum possible correlations. The function `R.bounds` replaces impossibly large or small values with maximally large or small values repectively.

Value

Functions return a scalar defining minimum or maximimum possible correlations. See Aho et al. (2023).

Author(s)

Ken Aho

References

Aho, K., Derryberry, D., Godsey, S. E., Ramos, R., Warix, S., Zipper, S. (2023) The communication distance of non-perennial streams. EarthArXiv https://eartharxiv.org/repository/view/4907/

Examples

```
min_r(0.6, 0.9)
max_r(0.1, 0.2)

x1 <- rep(c(1,0),5)
x2 <- c(rep(1,7), rep(0,3))
x3 <- c(rep(1,3), rep(0,7))
R <- cor(cbind(x1, x2, x3))
R.bounds(c(0.5, 0.7, 1), R)
```

`size.intact.to.sink` *Size of intact network that feeds into the sink or a particular node*

Description

The length of the subgraph network that ends (feeds into) a particular node, e.g., the sink. For a weighted graph, the sum of the weights of the subgraph are given. Thus, if weights are stream lengths the function will give the stream length of the portion of the intact stream network that feeds into a particular node.

Usage

```
size.intact.to.sink(G, sink = NULL)

size.intact.to.node(G, node = NULL)
```

Arguments

`G`	A graph object of class "igraph", see `graph_from_literal`
`sink`	The sink node of `G`.
`node`	A node of interest.

Value

Returns the size of the graph or subgraph that includes a node of interest.

Author(s)

Ken Aho, Gabor Csardi wrote several important function components including `subgraph`.

Examples

```
# Murphy Cr. data, no arc from M1799 to M1719
G <- graph_from_literal(IN_N --+ M1984 --+ M1909, IN_S --+ M1993 --+ M1951,
M1951 --+ M1909 --+ M1799, M1719 --+ M1653 --+ M1572 --+ M1452 --+ M1377,
M1377 --+ M1254 --+ M1166 --+ M1121 --+ M1036 --+ M918 --+ M823 --+ M759,
M759 --+ M716 --+ M624 --+ M523 --+ M454 --+ M380 --+ M233 --+ M153 --+ M91,
M91 --+ OUT)
```

```
data(mur_coords) # coordinate data
spatial.plot(G, mur_coords[,2], mur_coords[,3], names =  mur_coords[,1])

data(mur_lengths) # segment length data

lengths_new <- mur_lengths[-7,] # Drop M1799 -> M1719 arc length
E(G)$weight <- lengths_new[,2]
size.intact.to.sink(G, sink = "OUT")
size.intact.to.node(G, node = "OUT")
```

`spath.lengths` *Shortest path lengths and number of paths*

Description

The function `spath.lengths` calculates path lengths from all possible nodes to or from a designated node, i.e., the shortest in-paths and out-paths repsectively. Weighted path length are possible, including weighted path lengths based on field-observed instream arc lengths (see Examples). This results in "actual" path lengths in observed units. The function `n.tot.paths` calculates the total number of paths beginning or ending at all nodes in a graph, based on exponention of the the adjacency matrix.

Usage

```
spath.lengths(G, node = NULL, mode = "in", ignore.inf = TRUE)

n.tot.paths(G, mode = "in", sink = NULL)
```

Arguments

`G`	Graph of class "igraph". See `graph_from_literal`
`node`	Designated node.
`mode`	One of `"in"` or `"out"`. The former gives in-paths, whereas the latter gives out-paths.
`ignore.inf`	Logical. Whether infinite distances are to be ignored. By default `ignore.inf = TRUE`, allowing impossible upstream distances to be ignored in stream DAGs.
`sink`	Name of sink node.

Value

Lengths of paths to a node of interest.

Author(s)

Ken Aho , Gabor Csardi wrote `distances`

Examples

```
data(mur_lengths)
mur <- streamDAGs("mur_full")
n.tot.paths(mur)

spath.lengths(mur, "M1653")
E(mur)$weight <- mur_lengths[,2] # weighted (actual in-stream lengths in meters)
spath.lengths(mur, "M1653")
```

---

spatial.plot *Spatial plot of an igraph object or stream shapefile*

Description

Makes a spatial plot of a igraph object or stream shapefile, given nodal coordinates and node IDs.

Usage

```
spatial.plot(G, x, y, names = NULL,
                          plot = TRUE,
                          col = "lightblue",
                          cex.text = .4, cex = 1,
                          arrow.col = "lightblue", arrow.lwd = 1,
                          plot.bg = "white", pch = 21,
                          pt.bg = "orange", grid.lwd = 2,
                          plot.dry = FALSE,
                          col.dry = gray(.7),
                          cex.dry = 1, pch.dry = 19,
                          arrow.col.dry = gray(.7), arrow.lwd.dry = 1,
                          cnw = NULL, xlim = NULL, ylim = NULL,
                          arrow.warn = TRUE, ...)


spatial.plot.sf(x, y, names, shapefile = NULL, cex = 1, arrow.col = "lightblue",
                arrow.lwd = 1, pch = 21, pt.bg = "orange")
```

Arguments

`G`	Graph object, see `graph_from_literal`.
`x`	X-coordinates of nodes.
`y`	Y-coordinates of nodes.
`names`	Names of nodes, must use the same names as `G` and correspond to the order of coordinates in `x` and `y`.
`plot`	Logical. Create plot?
`shapefile`	Shapefile object brought in using library *sf*

col	point symbol color.
cex.text	Character expansion for node labels in plot; cex.text = 0 suppresses labels.
cex	Chahracter expnansion of point symbols.
arrow.col	Color of plot arrows.
arrow.lwd	Arrow line width.
plot.bg	Background color of plot.
pch	Plotting character.
pt.bg	Background color for plotting character.
grid.lwd	Grid line width; grid.lwd = 0 suppresses grid.
plot.dry	Logical. Should "dry" nodes, i.e., nodes in names (and x and y) that are not also in G be plotted?
col.dry	Color of "dry" nodes in plot.
cex.dry	Symbol sizer of "dry" nodes in plot.
pch.dry	Plotting character (symbol) of "dry" nodes in plot.
arrow.col.dry	Arrow color for "dry" arcs. Dry arrow rendering requires cnw designation (see Examples).
arrow.lwd.dry	Arrow line width for "dry" arcs. Dry arrow rendering requires cnw designation (see Examples).
cnw	Complete network spatial.plot object.
xlim	A numeric vector of length 2, giving the lower and upper *y*-axis limits.
ylim	A numeric vector of length 2, giving the lower and upper *x*-axis limits.
arrow.warn	Logical. The function arrows omits arrowheads (with a warning) for any arrow of length less than 1/1000 inch. To elimiate this warning (which may occur for nearby nodes) specify arrow.warn = FALSE.
...	Other arguments to plot

Details

The function spatial.plot makes a plot of a stream DAG, showing arc flow directions to and from spatial node locations. The function can also be used to identify node and arc arrow coordinates for plotting (see Examples). The function spatial.plot.sf can create a spatially explicit graph from a stream shapefile with the stream outlay under a ggplot framework (see Examples). The function spatial.plot can be used to distinguish dry and wet nodes and arcs) (see Examples).

Value

A plot and an invisible list containing the x and y coordinates of nodes: the objects $x and $y, respectively, and the x and y coordinates of start and end points of arc arrows:the objects $x0, $y0, $x1, and $y1, respectively.

Author(s)

Ken Aho

Examples

```
G <- graph_from_literal(IN_N --+ M1984 --+ M1909, IN_S --+ M1993,
M1993 --+ M1951 --+ M1909 --+ M1799 --+ M1719 --+ M1653 --+ M1572 --+ M1452,
M1452--+ M1377 --+ M1254 --+ M1166 --+ M1121 --+ M1036 --+ M918 --+ M823,
M823 --+ M759 --+ M716 --+ M624 --+ M523 --+ M454 --+ M380 --+ M233 --+ M153,
M153 --+ M91 --+ OUT)

data(mur_coords)

x <- mur_coords[,2]
y <- mur_coords[,3]
names <- mur_coords[,1]
spatial.plot(G, x, y, names)

# using shapefiles


library(ggplot2); library(sf); library(ggrepel)
mur_sf <- st_read(system.file("shape/Murphy_Creek.shp", package="streamDAG"))
g1 <- spatial.plot.sf(x, y, names, shapefile = mur_sf)

# modify ggplot
g1 + theme_classic()



#-- Distinguishing wet and dry arcs and nodes --#

data(mur_node_pres_abs) # STIC H2O presence/absence
npa <- mur_node_pres_abs[650,][,-1] # STC data from 8/9/2019 22:30
G1 <- delete.nodes.pa(G, npa) # delete nodes based STIC data

# Example 1 (only show wet nodes and arcs with associated wet nodes)
spatial.plot(G1, x, y, names)

# Example 2 (show wet nodes and arcs with associated wet nodes, and dry nodes)
spatial.plot(G1, x, y, names, plot.dry = TRUE)

# Example 3 (show wet nodes and arcs wet node arcs, and underlying network)
entire <- spatial.plot(G, x, y, names, plot = FALSE)
spatial.plot(G, x, y, names, plot.dry = TRUE, cnw = entire)


#-- Animation: drying of Johnson Draw drainage --#

jd_graph <- streamDAGs("jd_full")
data(AIMS.node.coords)
jd_coords <- AIMS.node.coords[AIMS.node.coords$site == "JD",]
jd_coords <- jd_coords[jd_coords$STIC_inferred_PA,]
data(jd_node_pres_abs)

pb = txtProgressBar(min = 1, max = 250, initial = 1, style = 3)
```

```
times <- round(seq(1,50322, length = 250),0)

for(i in 1:250){
  dev.flush()
  jd_sub <- delete.nodes.pa(jd_graph,
                            jd_node_pres_abs[times[i],][-1],
                            na.response = "treat.as.1")
  spatial.plot(jd_sub,
               x = jd_coords[,3],
               y = jd_coords[,2],
               names = jd_coords[,1],
               ylim = c(43.122, 43.129),
               xlim = c(-116.8, -116.775),
               plot.dry = TRUE, main = jd_node_pres_abs[,1][times[i]],
               xlab = "Longitude", ylab = "Latitude")
  dev.hold()
  Sys.sleep(.05)
  setTxtProgressBar(pb, i)
  }
```

---

stream.order *Strahler or Shreve stream order of a stream DAG*

Description

The function `stream.order` calculates Strahler or Shreve number for each each in a stream DAG. The function `sink.G` is a utility algorithm that subsets the graph if the sink node is part of a subgraph that is disconnected from other nodes.

Usage

```
sink.G(G, sink = NULL)

stream.order(G, sink = NULL, method = "strahler")
```

Arguments

`G`	Graph object of class "igraph", see: See `graph_from_literal`.
`sink`	Sink node from `G`.
`method`	One of `"strahler"` or `"shreve"`.

Details

Strahler stream order (Strahler 1957) is a "top down" system in which first order stream sections occur at the outermost tributaries. A stream section resulting from the merging of tributaries of the same order will have a Strahler number one unit greater than the order of those tributaries. A stream section resulting from the merging of tributaries of different order will have the Strahler stream order of the tributary with the larger Strahler number. Under Shreve stream order, (Shreve 1966) a stream section resulting from the merging of tributaries will have an order that is the sum of the order of those tributaries.

The function can currently only handle graphs with confluences (which, as noted above, serve to define the stream order) and simple islands (those without sub-islands and those whose downstream endpoint does not occur at a join). Under the current version, islands will not change the order of a reach.

Value

Returns Stahler or Shreve numbers for each stream DAG node.

Note

May be slow for extremely large and complex streams due to a reliance on loops.

Author(s)

Ken Aho

References

Shreve, R. L. (1966). Statistical law of stream numbers. *The Journal of Geology, 74*(1), 17-37.

Strahler, A. N. (1952). Hypsometric (area-altitude) analysis of erosional topology. *Geological Society of America Bulletin, 63* (11): 1117-1142

Examples

```
stream.order(G = streamDAGs("konza_full"), sink = "SFM01_1", method = "strahler")

stream.order(G = streamDAGs("konza_full"), sink = "SFM01_1", method = "shreve")
```

`streamDAGs` *Stream DAG datasets*

Description

The function contains a number of stream direct acyclic graph datasets written in *igraph* format. See: `graph_from_literal`. Many of the graphs were based on sampling regimes for the National Science Foundation Aquatic Intermittency Effects on Microbiomes in Streams (AIMS) project.

Usage

```
streamDAGs(graph = c("dc_piezo_full", "dc_full", "gj_full16", "gj_synoptic_2023",
"gj_full", "jd_piezo_full","jd_full", "konza_full", "KD0521", "KD0528", "KD0604",
"mur_full", "td_full", "wh_full", "pr_full"))
```

Arguments

graph — Currently, one of "dc_piezo_full", "dc_full", "gj_full16", "gj_full16", "gj_full", "gj_synoptic_2023", "jd_piezo_full", "jd_full", "konza_full", "KD0521", "KD0528", "KD0604", "mur_full", "pr_full", "td_full", or "wh_full".

Details

Currently, the following graph options exist:

1. "dc_piezo_full" codifies the Dry Creek stream network in southwestern Idaho for both STIC (Stream Temperature, Intermittency, and Conductivity) and piezometer locations (outlet coordinates: 43.71839°N, 116.13747°W).
2. "dc_full" codifies the Dry Creek stream network in southwestern Idaho but only for STICs, not piezometers, (outlet coordinates: 43.71839°N, 116.13747°W). This graph can be used in conjunction the dataset: dc_node_pres_abs. Network spatial coordinates can be obtained from AIMS.node.coords using: AIMS.node.coords$site == "DC".
3. "gj_full16" codifies nodes established at the Gibson Jack drainage in southeast Idaho, as defined in 2016 (outlet coordinates: 42.767180°N, 112.480240°W).
4. "gj_full" codifies nodes established at the Gibson Jack drainage in southeast Idaho, by the the AIMS team for seasonal sampling in 2022-2023 (outlet coordinates: 42.767180°N, 112.480240°W). The graph can be used in conjunction with the datasets gj_node_pres_abs and gj_lengths. Network spatial coordinates can be obtained from AIMS.node.coords using: AIMS.node.coords$site == "GJ".
5. "gj_synoptic_2023" codifies nodes established at the Gibson Jack drainage in southeast Idaho by the AIMS team during synoptic sampling in 2023, includes piezometers and additional sites to those sampled in "gj_full" (outlet coordinates: 42.767180°N, 112.480240°W).
6. "jd_piezo_full" codifies the Johnson Draw stream network in southwestern Idaho for both STC and and piezometer locations (outlet coordinates: 43.12256°N, 116.77630°W).
7. "jd_full" codifies the Johnson Draw stream network in southwestern Idaho, but only for STICs, not piezometers (outlet coordinates: 43.12256°N, 116.77630°W). This graph can be used in conjunction the datasets: jd_node_pres_abs and jd_lengths. Network spatial coordinates can be obtained from AIMS.node.coords using: AIMS.node.coords$site == "JD".
8. "konza_full" provides codification of a complete intermittent stream network of Konza Prairie in the northern Flint Hills region of Kansas (outlet coordinates: 39.11394°N, 96.61153°W). The network can be used in conjuction with the datasets kon_coords and kon_lengths.
9. Options "KD0521","KD0528", and "KD0604" provide networks for Konza Prairie at 05/21/2021 (before spring snow melt), 05/28/2021 (during spring snow melt) and 06/04/2021 (drying following snow melt), respectively.
10. "mur_full" is an *igraph* codification of the complete Murphy Creek dataset from the Owyhee Mountains in SW Idaho (outlet coordinates: 43.256°N, 116.817°W) established in 2019 by Warix et al. (2021), also see Aho et al. (2023). The network can be used in conjunction with the datasets: mur_coords, mur_lengths, and mur_node_pres_abs.
11. "pr_full" codifies the Painted Rock stream network in northern Alabama (outlet coordinates: 34.96867°N, 86.16544°W). Network spatial coordinates can be obtained from AIMS.node.coords using: AIMS.node.coords$site == "PR".

12. "td_full" codifies the Talladega stream network in central Alabama (outlet coordinates: 33.76218°N, 85.59552°W). Network spatial coordinates can be obtained from `AIMS.node.coords` using: `AIMS.node.coords$site == "TD"`.
13. "wh_full" codifies the Weyerhauser stream network in western Alabama (outlet coordinates: 32.98463°N, 88.01227°W). Network spatial coordinates can be obtained from `AIMS.node.coords` using: `AIMS.node.coords$site == "WH"`.

Value

Returns a graph object of class *igraph*.

Author(s)

Ken Aho, Rob Ramos, Rebecca L. Hale, Charles T. Bond, Arya Legg

References

Aho, K., Derryberry, D., Godsey, S. E., Ramos, R., Warix, S., & Zipper, S. (2023). The communication distance of non-perennial streams. *EarthArvix* doi: 10.31223/X5Q367

Warix, S. R., Godsey, S. E., Lohse, K. A., & Hale, R. L. (2021), Influence of groundwater and topography on stream drying in semi-arid headwater streams. *Hydrological Processes, 35*(5), e14185.

Examples

```
streamDAGs("mur_full")
```

Index