# Package: dagR (via r-universe)

September 18, 2024

**Type** Package

**Title** Directed Acyclic Graphs: Analysis and Data Simulation

**Version** 1.2.1

**Date** 2022-10-08

**Author** Lutz P Breitling

**Maintainer** Lutz P Breitling <`l.breitling@posteo.de`>

**Description** Draw, manipulate, and evaluate directed acyclic graphs and simulate corresponding data, as described in International Journal of Epidemiology 50(6):1772-1777.

**Imports** graphics, stats, utils

**Suggests** dagitty(>= 0.3-1)

**License** GPL-2

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-10-09 09:20:02 UTC

## Contents

---

`dagR-package` *Directed Acyclic Graphs: Analysis and Data Simulation*

---

## Description

The package dagR contains a couple of functions to draw, manipulate and evaluate directed acyclic graphs (DAG), with a focus on epidemiologic applications, namely the assessment of adjustment sets and potentially biasing paths. The functions for finding and evaluating paths essentially implement the graphical algorithms outlined in Greenland (1999).

When using this package for your work, please cite Breitling (2010) and/or Breitling et al. (2022).

For motivations to use this package in epidemiology teaching and methodological research, please refer to Duan et al. (2022).

*Note: As spelled out in the license, this suite of functions comes without any warranty, and cautious use is strongly advised. Although testing was carried out as meticulously as possible, it must be expected that bugs or errors remain, in particular in the early versions of the package. Please report any problems, concerns, but also suggestions for improvements or extensions to the author.*

Important additions in future versions could be e.g. improved drawing routines with better formatting of alternative node symbols in the DAG (taking into account the string length) and algorithms with intelligent/efficient search for minimal adjustment sets.

## Details

| | |
|---|---|
| Package: | dagR |
| Type: | Package |
| Version: | 1.2.1 |
| Date: | 2022-10-09 |
| License: | GPL-2 |
| LazyLoad: | yes |

`dag.init` is used for setting up DAGs. See the code of the functions `demo.dag0` to `demo.dag6` for example code. To adjust and/or evalute DAGs for biasing paths, use `dag.adjust`, `dag.draw` for drawing a DAG. `dag.search` uses `brute.search` to evaluate all possible adjustment sets, allowing the identification of minimal sufficient adjustment sets using `msas`. `dag.sim` simulates data (normally distributed or binary) according to the causal structure given by a DAG object.
In version 1.2.0, generic S3 methods (print, plot, summary) for dagR-DAGs were implemented, but the original functions `summary_dagRdag` to summarize and `dag.draw` to plot a DAG object were preserved for backwards compatibility. Export functions to other packages were added upon a reviewer request.
Several helper functions currently are not hidden and should later be made internal.

*Please see the NEWS file for version changes and known open issues.*

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777 <doi: 10.1093/ije/dyab167>.
Duan C, Dragomir AD, Luta G, Breitling LP (2022). Reflection on modern methods: Understanding bias and data analytical strategies through DAG-based data simulations. Int J Epidemiol

50(6):2091-2097 <doi: 10.1093/ije/dyab096>.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.

---

## `add.arc` *Add an arc to a DAG.*

### Description

Conveniently add an arc to an existing DAG.

### Usage

```
add.arc(dag, arc, type = 0)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to which an arc should be added. |
| `arc` | A vector of length 2, indicating from which node (first element) to which node (second element) the arc is to go. Note: the node numbering follows the numbering of the existing DAG (as shown in `dag.draw` with option `numbering=T`), not the numbering of `dag.init`. |
| `type` | 0 (=default) for a directed arc, 1 for an undirected association. |

### Value

A DAG with the arc (and corresponding `arc.type`) added, and with the path-related variables (`paths`, `pathsN`, `path.status`, `searchType`, `searchRes`) removed.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`rm.arc`, `add.node`, `rm.node`

---

`add.node` *Add a node to an existing DAG.*

---

## Description

Conveniently adds a node to an existing DAG, inserting its coordinates and label before the outcome node. Also updates the arcs correspondingly.

## Usage

```
add.node(dag, name = "unknown", type = 1, x = NA, y = NA)
```

## Arguments

| | |
|---|---|
| `dag` | The DAG to which the node is to be added. |
| `name` | Label for the node (defaults to "unknown"). |
| `type` | Type of node (1=covariable, 2=unknown); defaults to 1. |
| `x` | X coordinate for the node position. |
| `y` | Y coordinate for the node position. |

## Details

If no x and y coordinates are provided, the function places the node in an arbitrary position, slightly different with each additional node, so that one can more easily replace the nodes afterwards using `dag.move`.

## Value

A DAG with the new node added.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## See Also

`rm.node`, `add.arc`, `rm.arc`

---

## `addAngle` *Sum up two radian angles.*

---

### Description

Adds two radian angles together and applies modulus 2*pi. This is internally called by smoothArc, though hardly needed.

### Usage

```
addAngle(a, b)
```

### Arguments

| | |
|---|---|
| `a` | Angle 1 in radian. |
| `b` | Angle 2 in radian. |

### Value

numeric value [0, 2*pi).

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`smoothArc`

---

## `allCombs` *Create all combinations of the elements of a vector.*

---

### Description

Creates a matrix with all combinations of 1 to all elements of the vector provided. Elements to occur in all combinations can be specified. This is internally called by `brute.search`.

### Usage

```
allCombs(x, force = c(), trace = FALSE)
```

### Arguments

| | |
|---|---|
| `x` | A vector of elements of which combinations are to be formed. |
| `force` | A vector of elements that are supposed to occur in each combination. |
| `trace` | A boolean indicating if some output should be printed (TRUE) or not (FALSE=default). |

### Value

A matrix with one combination per row. For the shorter combinations, the columns to the right are filled up with NA.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`brute.search`

---

## `angle` *Calculate radian angle of line between two points.*

### Description

Calculates the radian angle of the line connecting two points. Internally called by `smoothArc`.

### Usage

```
angle(A, B)
```

### Arguments

| | |
|---|---|
| `A` | Vector of length two indicating the coordinates of the first point. |
| `B` | Vector of length two indicating the coordinates of the second point. |

### Value

A numeric value [0, 2*pi).

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`smoothArc`, `addAngle`

---

## `anglePoint` *Calculate coordinates at specific angle and distance.*

### Description

Calculates the coordinates of the point that is at a specific radian angle in a specific distance from a source point. Internally called by `smoothArc`.

### Usage

```
anglePoint(A, angl, len)
```

### Arguments

| | |
|---|---|
| `A` | Vector of length two with the coordinates of the source point. |
| `angl` | Radian angle indicating into which direction the new point is to be calculated. |
| `len` | The distance at which the new point is situated from the source point. |

### Value

A vector of length two with the coordinates of the new point.

### Note

Another pretty superfluous helper function...

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`smoothArc`

---

## `assoc.exists` *Check if association between two DAG nodes exists.*

### Description

Checks if an association between two DAG nodes already exists, i.e. does not need to be introduced when adjusting for a shared child etc. Internally called by `dag.adjustment`.

### Usage

```
assoc.exists(dag, a, b)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be dealt with. |
| `a` | First node. |
| `b` | Second node. |

### Value

A boolean indicating whether or not an association between first node and second node already exists.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

---

## `brute.search` *Evaluate all possible adjustment sets of a DAG.*

### Description

Evaluates all adjustment sets of a DAG, optionally including adjustment sets including "unknown" nodes. If the DAG has a non-empty adjustment set, only adjustment sets including these adjustment variables are evaluated.

### Usage

```
brute.search(dag, allow.unknown = FALSE, trace = TRUE, stop = 0)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be evaluated. |
| `allow.unknown` | Boolean indicating "unknown" nodes should be featured in the adjustment sets to be evaluated (TRUE) or not (FALSE=default). |
| `trace` | Boolean indicating if some output should be produced (TRUE=default). |
| `stop` | If =0, all eligible adjustment sets are evaluated. If =1, evaluations are stopped after the first sufficient adjustment set has been evaluated. Defaults to 0. |

### Value

A dataframe with the first columns (`X1..Xn`) indicating the variables in the respective adjustment set evaluated. The column `totalPaths` indicates the number of paths found when adjusting for the respective set, and `openPaths` indicates the number of biasing paths.

## Note

The output produced by `brute.search` allows to manually identify sufficient and minimal sufficient adjustment sets, which in the future should preferably be done by a helper summary function. The evaluation of a complicated DAG like `demo.dag2` can take quite some time, and future functions should either employ more intelligent algorithms to search specifically for sufficient sets, or they should allow e.g. the evaluation of adjustment sets of specific sizes.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

---

`dag.adjust` *Adjust an existing DAG for covariables.*

---

## Description

Looks for associations introduced by adjusting for the covariables specified, then looks for biasing paths, and finally evaluates these paths.

## Usage

```
dag.adjust(dag, A = c())
```

## Arguments

| | |
|---|---|
| `dag` | The DAG to be adjusted (or evaluated). |
| `A` | Vector indicating the adjustment set. *The numbering is according to the nodes vector of the DAG, which is shown e.g. in the legend of a DAG drawn by* `dag.draw`*. This numbering is different (+1) from the one used in* `dag.init`*, because the nodes vector also contains the exposure at position 1 (in contrast to the covariables vector used in* `dag.init`*)!* |

## Details

If the adjustment set is empty, the function only looks for biasing paths and evalutes these.

## Value

A DAG with the adjustment set A, and possibly with additional associations introduced by adjustment, biasing paths found, and the status of these.
If adjustment set is not empty, `searchType` and `searchRes` are set to `NULL`.

## Note

CAVE: Do not apply this to an already adjusted DAG, since this might not be handled appropriately (see documentation of `dag.adjustment` called by `dag.adjust`).

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.

### See Also

`dag.adjustment`, `find.paths`, `eval.paths`

---

## `dag.adjustment` *Adjust a DAG for one or more variables.*

### Description

Identifies the associations introduced by adjustment for the variables specified, and returns the DAG with these associations added. Note that this is called internally by `dag.adjust`, which makes sure that biasing paths are looked for and evaluted afterwards. Thus, `dag.adjustment` should 1.) *not* be called directly, and 2.) *not* be called on an already adjusted DAG!

### Usage

```
dag.adjustment(dag, A=NULL)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be adjusted. |
| `A` | The adjustment set to be applied. |

### Details

The adjustment set A specified when calling `dag.adjustment` overrules the adjustment variables that are present in the DAG. To keep these in the adjustment set, one has to add them to A.

### Value

A DAG with A as the adjustment set and the associations introduced by adjustment for A added to the DAG.

### Note

You should *not* use `dag.adjustment` on an already adjusted DAG, since it cannot identify associations that had been introduced by the earlier adjustment. If the new adjustment set does not include the adjustment variables present in the first set, the new DAG might feature associations that actually only would be introduced when adjusting for the variables featured in the first but not second adjustment set.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`dag.adjust`, `find.paths`, `eval.paths`

---

## `dag.ancestors` *Identify ancestors of DAG nodes.*

### Description

This identifies those nodes in a DAG that are ancestors of the nodes specified, i.e. acc. to the model depicted by the DAG they causally precede those nodes. Internally called by `dag.adjustment` in the context of finding associations introduced by adjustment.

### Usage

```
dag.ancestors(dag, A)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be evaluated. |
| `A` | A vector of nodes for which ancestors are to be identified. |

### Value

A vector indicating which nodes are ancestors of those in A. Note that A actually is included at the beginning of the vector.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`dag.adjust`, `dag.adjustment`

---

## `dag.draw` *Draw a DAG.*

---

### Description

Draws a DAG defined in an object of class dagRdag (as of dagR version 1.2.0, the generic function plot.dagRdag can be used for this purpose, but dag.draw is maintained for backwards compatibility). The nodes are represented by 'C' (covariables; numbered with subscripts) and 'U' (unknown/unmeasured covariables; numbered with subscripts), 'X' and 'Y' (exposure and outcome, respectively). A legend presents the names of the nodes. The X->Y arc is marked with a questionmark as the relationship of interest. Adjusted variables are under- and over-lined. Undirected associations are drawn with dashed lines. If paths have been identified (and evaluated), these (and their status) are written next to the legend.

### Usage

```
dag.draw(dag, legend = TRUE, paths = TRUE, numbering = FALSE,
 p = FALSE, alt.symb = TRUE, noxy = 0, ...)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be drawn. |
| `legend` | Boolean indicating whether a node legend should be included. |
| `paths` | Boolean indicating whether paths (and their status) should be written. |
| `numbering` | Boolean indicating whether the arcs should be numbered in the DAG. |
| `p` | Boolean indicating whether the curving points of undirected associations should be drawn. |
| `alt.symb` | Boolean indicating if the alternative node symbols (dag$symbols) should be used. Note that especially the legends and paths will not be formatted nicely if these symbols are longer strings. |
| `noxy` | Integer to indicate if the X->Y should not be drawn (0=default; 1=no arc; 2=arc, but no question mark). |
| `...` | *Currently not used.* |

### Value

Returns the DAG (for whatever reason...).

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.

### See Also

`dag.letter`, `garrows`, `smoothArc`, `dag.legend`, `write.paths`

---

## `dag.init` *Set up a new DAG.*

### Description

Allows setting up a new DAG. See the `demo.dag0` to `demo.dag6` functions for some example specifications.

### Usage

```
dag.init(outcome = NULL, exposure = NULL, covs = c(), arcs = c(),
 assocs = c(), xgap = 0.04, ygap = 0.05, len = 0.1, y.name = NULL,
 x.name = NULL, cov.names = c(), symbols = NULL, ...)
```

### Arguments

| | |
|---|---|
| `outcome` | *Currently not used!* |
| `exposure` | *Currently not used!* |
| `covs` | Vector including an integer for each covariable to be in the DAG (1 for a "standard" covariable, 2 for an unknown/unmeasured one). |
| `arcs` | Vector of duplets of integers, in which nodes from which an arc or undirected association is to emenate are followed by those to which it is to point. To refer to the exposure, use 0, to refer to the outcome, use -1, to refer to covariables, use and element of 1:length(covs). |
| `assocs` | A vector of same length as covs, with 0 indicating directed arcs, 1 indicating undirected associations. |
| `xgap` | How much x space is to be left between arc ends and nodes when drawing? |
| `ygap` | How much y space is to be left between arc ends and nodes when drawing? |
| `len` | Length of arrow whiskers when drawing. |
| `y.name` | Label of outcome. |

| | |
|---|---|
| `x.name` | Label of exposure. |
| `cov.names` | Vector of covariable labels. |
| `symbols` | Vector of alternative node symbols. Longer symbols will not be formatted nicely. Note that the first element refers to the exposure, the following ones to the covariables, the last one to the outcome. |
| `...` | *Currently not used.* |

## Value

A DAG (objects of class `dagRdag`). Check out some of the demonstration DAGs for details. The DAG is actually a list object, with elements `cov.types` (the `covs` vector, with 0 put in front, and -1 at the end).; `x` and `y` (coordinates for drawing the nodes, initially set up more or less in a half-circle above the x->y arc); `arc` (the arcs, transformed into a matrix); `arc.type` (the `assocs` vector); `curve.x` and `curve.y` (if associations are featured, these provide the coordinates through which to curve); `xgap`, `ygap`, `len` (the respective drawing parameters); `symbols` (alternative node symbols); `version` (dagR version).

## Note

CAVE: The numbering of the covariables and arc coordinates is different here than in the functions later used on the DAG (e.g. `add.arc`, `dag.adjust`)! The functions generally work according to the indexing of the R objects that they handle. Whereas for `dag.init` the `n` covariable nodes are numbered `1:n`, the node vector of the resulting DAG will also contain the exposure node at the beginning and the outcome node at the end, i.e. it will go from `1:(n+2)` with the covariables at `2:n+1`. `summary_dagRdag` will show the latter numbering. *Example*: when adjusting for the first covariable, `dag.adjust` must be handed the adjustment set `A=2`, as the first covariable will occupy the second node (the first node is occupied by the exposure).

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.

## See Also

`dag.draw`

## Examples

```
#dag.init(covs = c(1, 1), arcs = c(0, 2, 1, 2, 1, 0, -1, 2))
```

---

## `dag.legend` *Write the legend in a DAG drawing.*

---

### Description

Lists the DAG symbols along with their names/labels below a DAG drawn.

### Usage

```
dag.legend(dag, lx = -0.15, ly = -0.075, alt.symb = TRUE)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG for which the legend is needed. |
| `lx` | X coordinate for repositioning legend. |
| `ly` | Y coordinate for repositioning legend. |
| `alt.symb` | Boolean indicating if the alternative node symbols (dag$symbols) should be used. Note that the formatting is not changed, i.e. longer symbols will not be formatted nicely. |

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`dag.draw`, `write.paths`

---

## `dag.letter` *Write a DAG node symbol.*

---

### Description

Writes the node symbols, 'X' and 'Y' for exposure and outcome, 'C' and 'U' (with consecutive subscripts) for known and unknown covariables. Since v1.1.2, alt.symb allows the use of custom node symbols. Unknownness is identified by either node name 'unknown' or covariable type '2' in the DAG object. Note that adjusted nodes are marked by bar and underline; this currently does not apply to those marked as unknown.

### Usage

```
dag.letter(dag, letter, x, y, alt.symb = TRUE)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG for which a node is to be written. |
| `letter` | The node that is of interest. |
| `x` | X position. |
| `y` | Y position. |
| `alt.symb` | Boolean indicating if custom symbols (dag$symbols) should be used if available. |

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`dag.draw`, `dag.legend`, `write.paths`

---

## `dag.letter2` *Return a DAG node symbol.*

### Description

Similar to dag.letter(), but returning a string to label a DAG node. Adjusted nodes are marked by a preceding underscore.

### Usage

```
dag.letter2(dag, letter, alt.symb)
```

### Arguments

| | |
|---|---|
| `dag` | The dagRdag object for which a node symbol is to be returned. |
| `letter` | The number of the node for which the symbol (often a single letter...) is to be returned. |
| `alt.symb` | If TRUE, the alternative node symbols of the DAG object will be used. |

### Value

A string containing the DAG letter or alternative symbol.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`dag.letter`

---

## `dag.move` *Interactively move a node or curving point in a DAG.*

---

### Description

This allows to reposition a node or association curving point of a DAG graphically. First, select a node or curving point by left-clicking close to it. Then reposition it to any other position by left-clicking. Once you are happy with the new position, right-click to exit.

### Usage

```
dag.move(dag)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be modified. |

### Value

The same DAG, but with the feature repositioned.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

---

## `dag.search` *Evaluate possible adjustment sets of a DAG.*

---

### Description

Currently, this simply is a wrapper for `brute.search`, which returns the input DAG with the results of `brute.search` and a string describing the search setup.

### Usage

```
dag.search(dag, type = "brute", allow.unknown = FALSE, trace = FALSE, stop = 0)
```

### Arguments

| | |
|---|---|
| `dag` | DAG to be evaluated. |
| `type` | Type of search to be performed. Currently, only =brute is possible. |
| `allow.unknown` | See `brute.search`. |
| `trace` | See `brute.search`. |
| `stop` | See `brute.search`. |

## Value

The DAG with components `searchType` and `searchRes` added.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## See Also

`brute.search`

---

## `dag.sim` *Simulate data based on a DAG.*

## Description

Simulates data according to a DAG object. This function may be replaced by dag.sim2 in the future.

## Usage

```
dag.sim(dag, b = rep(0, nrow(dag$arc)), bxy = 0, n,
            mu = rep(0, length(dag$x)),
        binary = rep(0, length(dag$x)),
         stdev = rep(0, length(dag$x)), naming = 2, seed = NA, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `dag` | The DAG object according to which data is to be simulated. |
| `b` | Vector of coefficients defining the direct effects of the DAG arcs. |
| `bxy` | Coefficient defining the direct effect of main exposure X on outcome Y. |
| `n` | Number of observations to be simulated. |
| `mu` | Vector of means that are to be simulated for the different DAG nodes. For binary nodes without an ancestor, the mean is taken as the prevalence to be simulated. For binary nodes with ancestors, the mean is similarly interpreted (see details in Value section). |
| `binary` | Vector indicating which nodes are to be continuous (=0) and binary (=1). |
| `stdev` | Vector of standard deviations for each node. For nodes without ancestors, continuous data are drawn from a Normal distribution with this standard deviation. For nodes with ancestors, this is the standard deviation of the residual noise that is added to the calculated observation values. |
| `naming` | If =2, the alternative DAG node symbols are used for naming the variables in the output dataframe. Otherwise, the output dataframe variables are named X1...Xn. |
| `seed` | Seed to initialize the random number generator. |
| `verbose` | If =TRUE, additional output is given during the simulation, in particular showing the different calculation steps. |

### Value

A dataframe with n (rows) observations featuring simulated data for each node (columns) in the DAG. Simulation steps: 1. simulate data for nodes i without ancestors, drawing from Normal distribution with mean mu[i] and stdev[i] (continuous node), or drawing from Bernoulli events with probability mu[i] (binary node). 2. simulate data for nodes i for which all ancestors already have been simulated by multiplying the ancestor values with the corresponding arc coefficients and summing them up, shifting the resulting values to the mean mu[i] specified for the currently simulated node (logit-transformed if binary), then adding noise drawn from a Normal distribution with mean 0 and standard deviation stdev[i], finally using the inverse logit of the resulting values as success probabilities for simulating binary data if node is binary.

### Note

Undirected arcs are ignored in these simulations!

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.
Duan C, Dragomir AD, Luta G, Breitling LP (2022). Reflection on modern methods: Understanding bias and data analytical strategies through DAG-based data simulations. Int J Epidemiol 50(6):2091-2097.

### See Also

`dag.sim2`

---

## `dag.sim2` *Simulate data based on a DAG.*

### Description

Simulates data according to a DAG object.

### Usage

```
dag.sim2(dag, b = rep(0, nrow(dag$arc)), bxy = 0, n,
                      distr  = rep(0, length(dag$x)),
                      mu     = rep(0, length(dag$x)),
                      stdev  = rep(0, length(dag$x)),
                      nu     = NA,
```

```
                lambda = NA,
                binary = NA,
                naming = 2, seed = NA, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `dag` | The DAG object according to which data is to be simulated. |
| `b` | Vector of coefficients defining the direct effects of the DAG arcs (on linear scale). |
| `bxy` | Coefficient defining the direct effect of main exposure X on outcome Y (on linear scale). |
| `n` | Number of observations to be simulated. |
| `distr` | 0 for Normal distribution continuous nodes,<br>1 for binary nodes simulated from logistic model,<br>1.1 for binary nodes simulated from logistic model (see mu),<br>2 for binary nodes simulated from linear risk difference model,<br>2.1 for binary nodes simulated from linear risk difference model (see mu) |
| `mu` | Vector of means that are to be simulated for the different DAG nodes:<br>For normally distributed continuous variables, the overall mean simulated.<br>For binary variables w/ distr=1 or distr=2, overall proportion of successes simulated.<br>For binary variables w/ distr=1.1 or distr=2.1, proportion of successes simulated in the reference category (sum of coef-weighted predictors =0). |
| `stdev` | Vector of standard deviations for each node.<br>For nodes without ancestors, continuous data are drawn from a Normal distribution with this standard deviation. For continuous nodes with ancestors, this is the standard deviation of the residual noise that is added to the calculated observation value.<br>If used on binary variables with ancestors, this would analogously add residual noise to the calculated predictor, diluting the direct effects. |
| `nu` | Not used. |
| `lambda` | Not used. |
| `binary` | For backwards compatibility: Vector indicating which nodes are to be continuous (=0) and binary (=1). If given, this is passed to argument "distr" and a warning is issued. |
| `naming` | If =2, the alternative DAG node symbols are used for naming the variables in the output dataframe. Otherwise, the output dataframe variables are named X1...Xn. |
| `seed` | Seed to initialize the random number generator. |
| `verbose` | If =TRUE, additional output is given during the simulation, in particular showing the different calculation steps. |

## Value

A dataframe with n (rows) observations featuring simulated data for each node (columns) in the DAG.

Simulation steps:
1. simulate data for nodes i without ancestors, drawing from Normal distribution with mean mu[i] and stdev[i] (continuous node), or drawing from Bernoulli events with probability mu[i] (binary node).
2. simulate data for nodes i for which all ancestors already have been simulated by multiplying the ancestor values with the corresponding arc coefficients and summing them up, shifting the resulting values to the mean mu[i] (exceptions: distr=1.1 or distr=2.1, as detailed in "mu" above) specified for the currently simulated node (logit-transformed if binary based on logistic model), then adding noise drawn from a Normal distribution with mean 0 and standard deviation stdev[i], finally using the resulting values (inverse logit, if binary based on logistic model) as success probabilities for simulating binary data if node is binary.

As the noise is added after shifting to the mean, the mean of the simulated data will not be exact. Also, the noise is added before calculating descendant nodes, i.e. it is sort of true inter-individual variation, rather than measurement error.

For the risk difference model, the success probability calculated by summing the weighted ancestors can easily be <0 (or >1). If this happens, the probability is set to 0 (or 1), and a warning is issued.

## Note

Undirected arcs are ignored in these simulations.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.
Duan C, Dragomir AD, Luta G, Breitling LP (2022). Reflection on modern methods: Understanding bias and data analytical strategies through DAG-based data simulations. Int J Epidemiol 50(6):2091-2097.

## See Also

`dag.sim`

---

## `dagR2dagitty` *Create dagitty code from a dagR DAG*

---

### Description

Translates a DAG as defined in a dagRdag object created by dagR into the dagitty package format. Node labeling follows the rules used for plotting dagRdag objects, but adjusted nodes are marked by a preceding underscore instead of under- and over-line.

### Usage

```
dagR2dagitty(x, alt.symb = TRUE, only.code = TRUE)
```

### Arguments

| | |
|---|---|
| `x` | The dagR DAG to be translated. |
| `alt.symb` | Boolean indicating if the alternative node symbols should be used. |
| `only.code` | If TRUE, a string with R dagitty function call is returned, which should be checked by the user (and possibly edited as required) before running it to create an equivalent dagitty DAG. If FALSE and the dagitty package has been installed and loaded, the dagitty function is called directly and the resulting dagitty DAG is returned. |

### Value

Either a string containing dagitty syntax to translate the dagR DAG into dagitty format, or a dagitty object.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.
https://cran.r-project.org/package=dagitty

### See Also

`dag.letter2`

---

## `demo.dag0` *Set up demo DAG #0.*

### Description

Initializes a simple DAG used during the dagR development phase.

### Usage

```
demo.dag0()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`demo.dag1`, `demo.dag2`, `demo.dag3`, `demo.dag4`, `demo.dag5`, `demo.dag6`

---

## `demo.dag1` *Set up demo DAG #1.*

### Description

Initializes a classical "M DAG" useful for demonstrating harmful adjustment. The DAG is motivated by figure 3 in Fleischer (2008) and also featured in Breitling (2010).

### Usage

```
demo.dag1()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Fleischer NL, Diez Roux AV (2008). Using directed acyclic graphs to guide analyses of neighbourhood health effects: an introduction. J Epidemiol Community Health 62:842-846.

### See Also

`demo.dag0`, `demo.dag2`, `demo.dag3`, `demo.dag4`, `demo.dag5`, `demo.dag6`

---

## `demo.dag2` *Set up demo DAG #2.*

### Description

Initializes a more complex DAG, motivated by Shrier (2008). This DAG was used to examine the performance of `brute.search` and has been featured in Breitling (2010).

### Usage

```
demo.dag2()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Shrier I, Platt RW (2008). Reducing bias through directed acyclic graphs. BMC Med Res Methodol 8:70

### See Also

`demo.dag0`, `demo.dag1`, `demo.dag3`, `demo.dag4`, `demo.dag5`, `demo.dag6`

---

## `demo.dag3` *Set up demo DAG #3.*

---

### Description

Initializes a DAG motivated by the manual for the software DAG v0.11 (Knüppel 2009). This DAG has been featured in Breitling (2010).

### Usage

```
demo.dag3()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Knüppel S (2009). DAG v0.11 documentation (Oct 21, 2009). `https://hsz.dife.de/dag/`

### See Also

`demo.dag0`, `demo.dag1`, `demo.dag2`, `demo.dag4`, `demo.dag5`, `demo.dag6`

---

## `demo.dag4` *Set up demo DAG #4.*

---

### Description

Initializes a miscellaneous DAG. What happens if you adjust for the exposure's child?

### Usage

```
demo.dag4()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`demo.dag0`, `demo.dag1`, `demo.dag2`, `demo.dag3`, `demo.dag5`, `demo.dag6`

---

## `demo.dag5` *Set up demo DAG #5.*

### Description

Initializes a miscellaneous DAG. What happens if you adjust for the outcome's child?

### Usage

```
demo.dag5()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`demo.dag0`, `demo.dag1`, `demo.dag2`, `demo.dag3`, `demo.dag4`, `demo.dag6`

---

## `demo.dag6` *Set up demo DAG #6.*

### Description

Initializes a miscellaneous DAG. What happens if you adjust for the collider?

### Usage

```
demo.dag6()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`demo.dag0`, `demo.dag1`, `demo.dag2`, `demo.dag3`, `demo.dag4`, `demo.dag5`

---

## `demo.dag7` *Set up demo DAG #7.*

### Description

Initializes a DAG motivated by the manual for the software DAG v0.11 (Kn\"uppel 2009). This DAG has been featured in Breitling (2010). The DAG is the same as DAG #3, but #7 demonstrates the use of alternative node symbols.

### Usage

```
demo.dag7()
```

### Value

Returns a DAG.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Kn\"uppel S (2009). DAG v0.11 documentation (Oct 21, 2009). https://hsz.dife.de/dag/

### See Also

`demo.dag3`

---

`distPoints` *Calculate distance between two points.*

---

## Description

Another rather superfluous helper function, internally used by `smoothArc`. Calculates the distance between two points.

## Usage

```
distPoints(A, B)
```

## Arguments

| | |
|---|---|
| `A` | Vector of length two, indicating x and y of first point. |
| `B` | Vector of length two, indicating x and y of second point. |

## Value

Distance between the two points.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

---

`eval.paths` *Evaluate potentially biasing paths in a DAG.*

---

## Description

This essentially implements the graphical algorithm described in Greenland (1999) to identify open "backdoor" (or not strictly backdoor, but potentially biasing) paths in a DAG. Paths are identified as being 'open', 'blocked by collider', or 'blocked by adjustment'. If both latter conditions apply, 'blocked by collider' is returned.

## Usage

```
eval.paths(dag)
```

## Arguments

| | |
|---|---|
| `dag` | A DAG to which `find.paths` has already been applied<br>(e.g. within `dag.adjust`). |

### Details

This function identifies a collider-blocked path as 'blocked by collider' even if it has been unblocked by adjusting for the collider. One could argue that this should not be the case. However, the biasing seems to be sufficiently represented in the DAG by the introduction of the association "jumping" the collider and potentially opening biasing paths.

### Value

A DAG with component `path.status` added.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.

### See Also

`dag.adjust`, `find.paths`

---

## `find.paths` *Find potentially biasing paths in a DAG.*

### Description

This identifies paths linking exposure and outcome in a DAG. Forward paths (including a directed arc emanating from the exposure) are *not* identified.

### Usage

```
find.paths(dag)
```

### Arguments

| | |
|---|---|
| `dag` | A DAG for which paths should be found. |

### Value

A DAG with components `pathsN` (number of paths identified) and `paths` (matrix with each row describing one path by indicating the arcs forming the path; ends with `NA` as some other function recognize the end of the path that way) added.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.

## See Also

`dag.adjust`, `eval.paths`

---

# `garrows` *Draw a directed arc in a DAG.*

## Description

Internally called by `dag.draw` for drawing directed arcs.

## Usage

```
garrows(x0, y0, x1, y1, xgap, ygap, len = 0.1)
```

## Arguments

| | |
|---|---|
| `x0` | X coordinate of origin. |
| `y0` | Y coordinate of origin. |
| `x1` | X coordinate of target node. |
| `y1` | Y coordinate of target node. |
| `xgap` | Space between node and arc ends on x axis. |
| `ygap` | Space between node and arc ends on y axis. |
| `len` | Length of arrow whiskers (default=0.1). |

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## See Also

`dag.draw`, `smoothArc`

---

## `inAngle` *Calculate angle between two arcs.*

---

### Description

Another rather superfluous helper function, calculating the radian angle between two radian angles. Internally called by `smoothArc`.

### Usage

```
inAngle(a, b)
```

### Arguments

| | |
|---|---|
| `a` | Radian angle 1. |
| `b` | Radian angle 2. |

### Value

Numeric in range from -pi to pi.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`smoothArc`

---

## `is.acyclic` *Check if a DAG actually is acyclic.*

---

### Description

This function checks for each node in a DAG whether backtracing arcs leading to it results in an "infinite recursion" error indicating that there actually is a cyclic part in the DAG (which then obviously seems not to be a DAG).

### Usage

```
is.acyclic(dag, maxSecs=NA)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG to be check. |
| `maxSecs` | maximum time before function aborts; |

## Value

A list with two elements. `acyclic` is a boolean indicating whether the DAG is acyclic (=TRUE) or contains a cyclic component (=FALSE). `nodewise` is a vector containing 1 boolean per node in the DAG, TRUE indicating that backtracing from this node does not lead to a cyclic component, FALSE indicating that backtracing from this node leads to a cyclic component.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

---

# `is.in` *Check if a specific numeric value occurs in a vector.*

## Description

Another trivial helper function, called internally by `eval.paths`. It checks whether the specified (numeric) value is part of a specified vector of (numeric) values.

## Usage

```
is.in(x, c = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A numeric value, for which the presence in a vector is to be checked. |
| `c` | A vector of numeric values. |

## Value

Boolean; TRUE if value is present, FALSE if not.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## See Also

`eval.paths`

## `is.unknown` *Check if a DAG node presents an unknown variable.*

### Description

Another helper function, internally used by `brute.search`. It checks whether the node specified is of type=2 or is named 'unknown'.

### Usage

```
is.unknown(x, dag)
```

### Arguments

| | |
|---|---|
| `x` | The node of interest. |
| `dag` | The DAG to be evaluated. |

### Value

TRUE if unknown (acc. to type or name), FALSE otherwise.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`brute.search`

## `msas` *Identify minimal sufficient adjustment sets.*

### Description

Evaluates DAG adjustment sets identified by a `dag.search` (or `brute.search`) for minimal sufficiency by counting for each sufficient adjustment set A how many smaller sufficient ones that are contained in A exist.

### Usage

```
msas(adjSets)
```

### Arguments

| | |
|---|---|
| `adjSets` | The `searchRes` component of a DAG (or the output of `brute.search`, which is used by `dag.adjust` to produce `searchRes`). |

## Value

A vector containing a `-1` for each insufficient adjustment set, and for sufficient ones the number of smaller sufficient ones contained in it.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.
Knüppel S, Stang A (2010). DAG Program: identifying minimal sufficient adjustment sets. Epidemiology 21(1):159.

## See Also

`viv`, `summary.dagRdag`

---

# `plot.dagRdag` *Function to draw a DAG*

## Description

Generic function to draw a directed acyclic graph in an object of class dagRdag. This essentially passes the DAG object to the function `dag.draw`, which is maintained for backwards compatibility.

## Usage

```
## S3 method for class 'dagRdag'
plot(x, y, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object of class dagRdag to be passed to `dag.draw`. |
| `y` | Currently not used. |
| `...` | Other arguments to be passed to `dag.draw`. |

## Details

For all available arguments, see documentation of `dag.draw`.

## Value

The DAG object is returned.

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Breitling LP, Duan C, Dragomir AD, Luta G (2022). Using dagR to identify minimal sufficient adjustment sets and to simulate data based on directed acyclic graphs. Int J Epidemiol 50(6):1772-1777.

## See Also

`dag.draw`

---

# `print.dagRdag` *Prints the raw contents of an object of class dagRdag.*

## Description

Generic function `print` code for class dagRdag. This uses the default `print` method for list objects and points the user to the availability of the more convenient `summary` method.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | An object of class dagRdag. |
| `...` | Other arguments passed to the print routine. |

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

---

## `rm.arc` *Remove an arc from a DAG.*

---

### Description

Conveniently remove an arc from an existing DAG.

### Usage

```
rm.arc(dag, arc)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG from which to remove the arc. |
| `arc` | A single integer, indicating which arc is to be removed (refering to the respective row of the `dag$arc` matrix). |

### Value

A DAG with the arc specified removed along with the corresponding attributes like arc types, curves, and path evaluation variables.

### Note

The numbering of the arcs can be visualized by applying `dag.draw` with the option "numbering=TRUE".

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`add.arc`, `add.node`, `rm.node`

---

## `rm.node` *Remove a node from a DAG.*

---

### Description

Conveniently remove a node from an existing DAG.

### Usage

```
rm.node(dag, node)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG from which to remove the node. |
| `node` | A single integer, indicating which node is to be removed. |

### Value

A DAG with the node specified removed, along with the corresponding attributes and dependent variables, i.e. arcs involving this node are also removed, and the numbering of the nodes (and their occurrence in arcs) is corrected accordingly.
Note: Search components (`searchType`, `searchRes`) of the DAG currently are generally set to NULL, even if no path is removed. This is for simplicity, because the node numbers would need to be changed eg. in the `searchRes` variables etc.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`add.node`, `rm.arc`, `add.arc`

---

## `smoothArc` *Draw an undirected assocation in a DAG.*

### Description

This draws a dashed connection between two points, curving it so that it goes through a third point. This is internally used by `dag.draw` to draw associations.

### Usage

```
smoothArc(A, B, C, res = 20, gap = 0.05, p = FALSE)
```

### Arguments

| | |
|---|---|
| `A` | Vector of length 2, providing xy coordinates of first point. |
| `B` | Vector of length 2, providing xy coordinates of second point. |
| `C` | Vector of length 2, indicating xy coordinates through which the association should be curved. |
| `res` | How smooth should the curve be drawn? |
| `gap` | How far from point A and B should the line end? |
| `p` | If TRUE, the point through which the curve goes is drawn (this is to allow better moving it with `dag.move`). |

### Note

In the version 1.0.1 distributed as online supplemental material with Breitling (2010), the function contains arbitrary default values used during development.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.

### See Also

`dag.draw`, `dag.move`

---

## `summary.dagRdag` *Summarize a DAG.*

### Description

Generic function `summary()` for class `dagRdag`.

### Usage

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

### Arguments

| | |
|---|---|
| `object` | An object of class `dagRdag`. |
| `...` | Currently not used. |

### Details

Summarizes according to what functions have been applied to the DAG. It does not itself call `dag.search` and the like. Exception: is calls `is.acyclic` (with `maxSecs=5`).
This function passes the `object` to `summary_dagRdag`, which is preserved for backwards compatibility.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic research. Epidemiology 10(1):37-48.
Kn\"uppel S, Stang A (2010). DAG Program: identifying minimal sufficient adjustment sets. Epidemiology 21(1):159.

---

# `summary_dagRdag` *Summarize a DAG.*

## Description

Generic function `summary()` working code for class `dagRdag`, which is used by package `dagR` from version 1.1.1 on. From version 1.2.0, `summary.dagRdag()` is available as a generic function, but summary_dagRdag is preserved for backwards compatibility.

## Usage

```
summary_dagRdag(dag)
```

## Arguments

| | |
|---|---|
| `dag` | An object of class `dagRdag`. |

## Details

Summarizes according to what functions have been applied to the DAG. It does not itself call `dag.search` and the like. Exception: is calls `is.acyclic` (with `maxSecs=5`).

## Author(s)

Lutz P Breitling <l.breitling@posteo.de>

## References

Breitling LP (2010). dagR: a suite of R functions for directed acyclic graphs. Epidemiology 21(4):586-587.
Greenland S, Pearl J, Robins JM (1999). Causal diagrams for epidemiologic reserach. Epidemiology 10(1):37-48.
Kn\"uppel S, Stang A (2010). DAG Program: identifying minimal sufficient adjustment sets. Epidemiology 21(1):159.

---

## `viv` *Is a numeric vector in another vector?*

---

### Description

Checks if all numeric elements of a vector occur also in another vector. It is internally used by `msas` to check if some adjustment set is contained in another one.

### Usage

```
viv(v1, v2)
```

### Arguments

| | |
|---|---|
| `v1` | The vector whose occurrence in v2 is to be checked. |
| `v2` | The vector in which v1 might occur. |

### Details

If a value occurs more than once in v1, it is counted as contained in v2 if it appears there once. An empty v1 (consisting only of NA) is considered to be contained in any v2.

### Value

TRUE if v1 occurs in v2, FALSE otherwise.

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`msas`

---

## `write.paths` *Write the paths into a DAG drawing.*

---

### Description

Writes the paths into a DAG drawing, using the symbols (’C’, ’U’, ’X’, ’Y’) used in the drawing, indicating directed arcs by ’<’ and ’>’, undirected ones by ’-’. Since version 1.1.2, alt.symb allow usage of custom node symbols, though multi-character symbols will not be formatted well. Adjusted variables are under- and over-lined. If the paths have been evaluated using `eval.paths`, the status are also written.

### Usage

```
write.paths(dag, px = 0.5, py = -0.06, alt.symb = TRUE)
```

### Arguments

| | |
|---|---|
| `dag` | The DAG that has been drawn. |
| `px` | An x coordinate to change the position of the path writing. |
| `py` | A y coordinate to change the position of the path writing. |
| `alt.symb` | Boolean indicating if alternative node symbols (dag$symbols) should be used. |

### Author(s)

Lutz P Breitling <l.breitling@posteo.de>

### See Also

`dag.draw`, `find.paths`, `eval.paths`, `dag.legend`

# Index