Package 'vistla'

November 27, 2025

Title Detecting Influence Paths with Information Theory

Version 2.1.2

Description Traces information spread through interactions between features, utilising information theory measures and a higher-order generalisation of the concept of widest paths in graphs. In particular, 'vistla' can be used to better understand the results of high-throughput biomedical experiments, by organising the effects of the investigated intervention in a tree-like hierarchy from direct to indirect ones, following the plausible information relay circuits.
Due to its higher-order nature, 'vistla' can handle multimodality and assign multiple roles to a single feature.

License GPL (>= 3)

BugReports https://gitlab.com/mbq/vistla/-/issues

URL https://gitlab.com/mbq/vistla

Language en-GB

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends R (>= 3.5.0)

Imports grid

NeedsCompilation yes

Author Miron B. Kursa [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7672-648X>)

Maintainer Miron B. Kursa <m@mbq.me>

Repository CRAN

Date/Publication 2025-11-27 11:10:02 UTC

Contents

`branches` *Extract all branches of the Vistla tree*

Description

Gives access to a list of all branches in the tree.

Usage

```
branches(x, suboptimal = FALSE)

## S3 method for class 'vistla'
as.data.frame(x, row.names = NULL, optional = FALSE, suboptimal = FALSE, ...)
```

Arguments

`x`	vistla object.
`suboptimal`	if TRUE, sub-optimal branches are included.
`row.names`	passed to `as.data.frame`.
`optional`	passed to `as.data.frame`.
`...`	ignored.

Value

A data frame collecting all branches traced by vistla. Each row corresponds to a single branch, i.e., edge between feature pairs. This way it is a triplet of original features, names of which are stored in `a`, `b` and `c` columns. For instance, path $I \to J \to K \to L \to M$ would be stored in three rows, for (a, b, c)=(I, J, K), (J, K, L) and (K, L, M). The width of a path (minimal ι value) between root and feature pair (b, c) is stored in the `score` column. `depth` stores the path depth, starting from 1 for pairs directly connected to the root, and increasing by one for each additional feature. Final column, `leaf`, is a logical path indicating whether the edge is a final segment of the widest path between root and c.

Note

Pruned trees (obtained with `prune` and using `targets` argument in the `vistla` call) have no suboptimal branches.

`cchain` *Synthetic continuous data representing a simple mediator chain*

Description

Chain is generated from an uniform variable X by progressively adding gaussian noise, producing a mediator chain identical to this of the `chain` data, i.e.,

$$Y \to M_1 \to M_2 \to M_3 \to M_4 \to T$$

The set consists of 20 observations, and is tuned to be easily deciphered.

Usage

```
data(cchain)
```

Format

A data set with six numerical columns.

`chain` *Synthetic data representing a simple mediator chain*

Description

Chain is generated from a simple Bayes network,

$$Y \to M_1 \to M_2 \to M_3 \to M_4 \to T$$

where every variable is binary. The set consists of 11 observations, and is tuned to be easily deciphered.

Usage

```
data(chain)
```

Format

A data set with six binary factor columns.

`collapse` *Collapse the vistla tree into a pairwise graph*

Description

Collapse the vistla tree into a pairwise graph

Usage

```
collapse(x, aggregate = c("max", "sum", "none"))
```

Arguments

`x`	vistla object or a vistla_hierarchy object to collapse.
`aggregate`	score aggregation mode. "max" is the maximal score for this edge over all paths in the tree. For raw vistla scores it means the score of the widest path this edge was a part of; for ensemble scores, it corresponds to the count of the most often appearing path with this edge. "sum" is the sum of scores. Makes little sense for raw vistla scores; for ensemble scores it corresponds to the total count of this edge over all paths in the ensemble. "none" returns a vector of scores over all paths, which can be processed anyhow the user desires.

Value

A pairlist representation of the graph resulting from the tree collapse. The result is a data frame with the following columns. `A` & `B` are the ends of the edge, in order where A is closer to root than B (interpretation depends on the `flow` parameter used in `vistla` invocation); `score` is the score aggregated according to the `aggregate` argument; finally `paths` is the count of paths which included this edge.

Examples

```
## Not run:
 data(junction)
 v<-vistla(Y~.,data=junction)
 collapse(v)

## End(Not run)
```

ensemble *Construct the value for the ensemble argument*

Description

Vistla can be run in the ensemble mode, in which tree is built multiple times, usually on a slightly modified input data. This mode can be triggered by passing a value to the ensemble argument of the vistla method. This function can be used to construct the proper value for this argument.

Usage

```
ensemble(n = 30, resample = TRUE, prune = 0)

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

Arguments

n	number of replications.
resample	if TRUE, a modified bootstrap is used; that is, algorithm draws as many objects as are in the original data, but with replacement, hence only about 63.2 If this argument is given a number, it is interpreted to randomly sample exactly this fraction of objects, without replacement. Fraction f of n objects is interpreted as round(n*f), but not less than 3 and no more than n-1. If FALSE, no resampling is done (vistla trees are just built using different random seeds.
prune	Minimal number of iterations in which certain branch must appear not be pruned during ensemble consolidation. Zero (default) means no pruning. Note that iomin and targets arguments of the base algorithm can also be used to control the size of the resulting consensus tree.
x	ensemble control value to print.
...	ignored.

Value

A vistla_ensemble_control object which can be passed to the vistla function.

flow *Construct the value for the flow argument*

Description

Vistla builds the tree by optimising the influence score over path, which is given by the iota function. The flow argument of the vistla function can be used to modify the default iota and some associated behaviours. This function can be used to construct the proper value for this argument.

Usage

```
flow(code, ..., from = TRUE, into = FALSE, down, up, forcepath)

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

Arguments

code	Character code of the flow parameter, like "fromdown". If given, overrides other arguments.
...	ignored.
from	if TRUE, paths must satisfy data processing inequality as going from the root.
into	if TRUE, paths must satisfy data processing inequality as going into the root.
down	if TRUE, subsequent features on the path must have lower mutual information with the root; by default, true when from is true but if both from and into are true. Can't be true together with up.
up	if TRUE, subsequent features on the path must have higher mutual information with the root; by default, true when into is true but if both from and into are true. Can't be true together with down.
forcepath	when neither up or down is true, vistla may output walks rather than paths, i.e., sequences of features which are not unique. Yet, when this argument is set to TRUE, additional condition is checked to forbid such self-intersections. One should note that this check is computationally expensive, though. By default true when both up and down are false.
x	flow value to print.

Value

A vistla_flow object which can be passed to the vistla function; in practice, a single integer value.

---

hierarchy *Extract the vertex hierarchy from the vistla tree*

Description

Traverses the vistla tree in a depth-first order and lists the visited vertices as a data frame.

Usage

```
hierarchy(x)
```

Arguments

x	vistla object.

Value

A data frame of a class `vistla_hierarchy`.

Note

This function effectively prunes the tree off suboptimal paths.

`junction` *Synthetic data representing a junction*

Description

Junction is a model of a multimodal agent, a variable that is an element of multiple separate paths. Here, these paths are $Y \to A_1 \to A_2 \to J \to A_3$ and $Y \to B_1 \to B_2 \to J \to B_3$, while J is the junction. The set consists of 50 observations.

Usage

```
data(junction)
```

Format

A data set with eight factor columns.

`mi_scores` *Extract mutual information score matrix*

Description

Produces a matrix S where S_{ij} is a value of $I(X_i; X_j)$. This matrix is always calculated as an initial step of the vistla algorithm and stored in the vistla object.

Usage

```
mi_scores(x)
```

Arguments

`x`	vistla object.

Value

A symmetric square matrix with mutual information scores between features and root.

`mle_coerce` *Basic discretisation of numerical features*

Description

One can use this function for a quick, ad hoc discretisation of numerical features in a data frame, so that it could be passed to `vistla` using the maximal likelihood estimation (mle, the default). This can be used to simulate legacy behaviour of vistla, which was to automatically perform such conversion with 10 equal-width bins. The non-numeric columns are left as they were, hence this function is idempotent and does nothing when given fully discrete data.

Usage

```
mle_coerce(x, bins = 3, equal = c("size", "width"))
```

Arguments

`x`	Data frame to be converted.
`bins`	Number of bins to cut each numerical column into.
`equal`	If given `"width"`, function performs cuts into bins of an equal width, which may thus contain substantially different number of objects. One the other hand, when given `"size"` (default), cuts are done according to quantiles, hence provide bins with approximately the same number of objects, yet with different widths. Both options are asymptotically equivalent when the distribution of a given column is uniform.

Value

A copy of `x`, in which numerical columns have been discretised.

Note

While convenient, this function does not necessary provide optimal quantisation of the data (in terms of future vistla performance); especially the bins parameter should be adjusted to the input data, either via optimisation or based on the known properties of the input or mechanisms behind it.

Examples

```
## Not run:
data(cchain)
vistla(Y~.,data=mle_coerce(cchain,3,"size"))

## End(Not run)
```

paths *List all paths*

Description

Executes path_to for all path possible targets and returns a list with the results.

Usage

```
paths(x, targets_only = !is.null(x$targets), detailed = FALSE)
```

Arguments

x	vistla or vistla_hierarchy object.
targets_only	if TRUE, only paths to targets are extracted. By default, turned on when x has targets, and off otherwise. Ignored when x is a vistla_hierarchy.
detailed	passed to path_to. If TRUE, suppresses default output and presents the same paths in a form of data frames featuring score.

Value

A named list with one element per leaf or target, containing the path between this feature and root, in a format identical to this used by the path_to function.

path_to *Extract a single path*

Description

Gives access to a vector of feature names over a path to a certain target feature.

Usage

```
path_to(x, target, detailed = FALSE)
```

Arguments

x	vistla or vistla_hierarchy object.
target	target feature name.
detailed	if TRUE, suppresses default output and presents the same paths as a data frame featuring score.

Value

By default, a character vector with names of features along the path from target into root. When detailed is set to TRUE and input is a vistla object, a data.frame in a format identical to this produced by branches, yet without the leaf column.

plot.vistla *Overview plot of the vistla tree*

Description

Plots a vistla tree, using layout derived by a Buchheim et al. extension of the standard Reingold-Tilford method. The tree root is placed on the left, while the paths extend to the right, with all branches of the same depth at the same horizontal coordinate. The path are sorted vertically, from strongest on top to weakest on the bottom. Link weight indicates, by default, the link's score. A feature name in parentheses indicates that is is only a way-point in a path to some other feature.

Usage

```
## S3 method for class 'vistla'
plot(
  x,
  ...,
  slant,
  circular,
  asp1 = FALSE,
  pmar = c(0.05, 0.05, 0.05, 0.05),
  edge_col = 1,
  edge_lwd = "scale",
  edge_lty = 1,
  label_text = function(x) x$name,
  label_border_col = 1,
  label_border_lty = function(x) ifelse(x$leaf, 1, 2),
  label_fill = "white"
)

## S3 method for class 'vistla_plot'
plot(x, ...)

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

Arguments

x	vistla, vistla hierarchy or vistla plot object.
...	ignored.
slant	arrange vertices in a slanted way. Can be given as a number, possibly negative, indicating the amount of slant, or as TRUE, for an auto value. No slant is applied when set to 0 or omitted.
circular	if given TRUE, switches to circular layout; alternatively, can be given two numbers, then the first one will be interpreted as an angle to fit the whole graph in (2*pi when using TRUE), and the second one as an initial angle offset (0 when

using TRUE), which can be used to rotate the whole graph around the root. Both angles are expected to be in radians. It is recommended to add asp=TRUE parameter to make this layout truly circular, otherwise lines of equal depth are going to be elliptical. When FALSE, linear layout is enforced.

asp1 if TRUE, scales on both axes are the same, like with asp=1 in base graphics.

pmar Specifies margins as a fraction of graph size; expects a 4-element vector, in standard R bottom-left-top-right order.

edge_col edge colour; can be given as vector, then mapping order adheres to the one in hierarchy object; please note that the edge towards first feature, the root, is not drawn, so the first element is effectively ignored. If given as a function, it is called on the internally generated extended hierarchy object, and the result is used as an aesthetic.

edge_lwd edge width; behaves similarly to edge_col, yet also accepts special value 'scale', which triggers default scaling of edge width to be proportional to score.

edge_lty edge line-type; behaves similarly to edge_col.

label_text vertex label text, feature name by default. Behaves similarly to edge_col.

label_border_col
vertex label border colour; behaves similarly to edge_col, can be set to 0 for no border.

label_border_lty
vertex label border line-type; behaves similarly to edge_col, can be set to 0 for no border.

label_fill vertex label fill colour; behaves similarly to edge_col, can be set to 0 for no fill.

Value

Grid object with the graph.

Note

The graph is rendered using the grid graphics system, in a manner similar to ggplot2; the output of the plot.vistla function is only a grid graphical object, while the actual plotting is done when this object is printed or plotted. Yet, said object can be used with other functions in the grid ecosystem for rendering into files, being edited, combined with other plots, etc.

References

"Drawing rooted trees in linear time" C. Buchheim, M. Jünger, S. Leipert. Software: Practice and Experience 36(6):651-665 (2006).

print.vistla_hierarchy *Print vistla objects*

Description

Utility functions to print vistla objects.

Usage

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

## S3 method for class 'vistla'
print(x, n = 7L, ...)
```

Arguments

x	vistla object.
...	ignored.
n	maximal number of paths to preview.

Value

Invisible copy of x.

prune *Prune the vistla tree*

Description

This function allows to filter out suboptimal branches, as well as weak ones or these not in particular paths of interest.

Usage

```
prune(x, targets, iomin, score)
```

Arguments

x	vistla object or a vistla_hierarchy object.
targets	a character vector of features. When not missing, all branches not on lying paths to these targets are pruned. Unreachable targets are ignored, while names not present in the analysed set cause an error.
iomin	a legacy name for score, valid only for vistla objects; passing a value to either of them works the same, but giving some values for both is an error.
score	a score threshold below which branches should be removed. When given, it effectively overrides the value of `iomin` or `ensemble(prune,...)` given to the `vistla` invocation; to this end, it can only be higher then the original value, since prune only modifies the output and cannot re-run the pathfinding.

Value

Pruned `x`; if both arguments are missing, this function still removes suboptimal branches.

Examples

```
## Not run:
 data(chain)
 v<-vistla(Y~.,data=chain)
 print(v)
 print(prune(v,targets="M3"))
 print(prune(v,score=0.3))

## End(Not run)
```

---

vistla *Influence path identification with the Vistla algorithm*

Description

Detects influence paths.

Usage

```
vistla(x, ...)

## S3 method for class 'formula'
vistla(formula, data, ..., yn)

## S3 method for class 'data.frame'
vistla(
  x,
  y,
  ...,
  flow,
```

```
  iomin,
  targets,
  estimator = c("mle", "kt"),
  verbose = FALSE,
  yn = "Y",
  ensemble,
  threads
)

## Default S3 method:
vistla(x, ...)
```

Arguments

x	data frame of predictors.
...	pass-through arguments, ignored.
formula	alternatively, formula describing the task, in a form `root~predictors`, which adheres to standard R behaviours. Accepts `+` to add a predictor, `-` to omit one, and `.` to import whole `data`. Use `I` to calculate new predictors. When present in `data`, response is getting omitted from predictors.
data	`data.frame` in context of which the formula will be executed; can be omitted when not using `..`
yn	name of the root (`Y` value), used in result pretty-printing and plots. Must be a single-element character vector.
y	vistla tree root, a feature from which influence paths will be traced.
flow	algorithm mode, specifying the iota function which gives local score to an edge of an edge graph. If in doubt, use the default, `"fromdown"`. Consult the documentation of the `flow` function for more information.
iomin	score threshold below which path is not considered further. The higher value the less paths are generated, which also lowers the time taken by the function. The default value of 0 turns of this filtering. The same effect can be later achieved with the `prune` function.
targets	a vector of target feature names. If given, the algorithm will stop just after reaching the last feature from this list, rather than after tracing paths to all targets. The same effect can be later achieved with the `prune` function, but restricting targets here saves computational time.
estimator	mutual information estimator to use. `"mle"` — maximal likelihood, requires all features to be discrete (factors or booleans). `"kt"` — Kendall transformation, requires all features to be either ordinal (numeric, integer or ordered factor) or bi-valued (two-level factors or booleans).
verbose	when set to `TRUE`, turns on reporting of the algorithm progress.
ensemble	used to switch vistla to the ensemble mode, in which a number of vistla models are built over permuted realisations of the input, and merged into a single consensus tree. Should be given an output of the `ensemble` function; as a short-cut, one can pass a single number, which will be interpreted as the number of replications with other ensemble parameter default. That is, `ensemble=30` is equivalent to `ensemble=ensemble(n=30)`. Permutations are applied before estimators.

`threads` number of threads to use. When missing or set to 0, vistla uses all available cores.

Value

Normally, the tracing results represented as an object of a class `vistla`. Use `paths` and `path_to` functions to extract individual paths, `branches` to get the whole tree and `mi_scores` to get the basic score matrix.

When `ensemble` argument is given, a hierarchy object with the scored being counts of times certain path was present among the replicated ensemble, possibly pruned.

Note

The ensemble mode is both faster and makes better use of multithreading than replicating vistla manually.

References

"Vistla: identifying influence paths with information theory" M.B. Kursa. Bioinformatics btaf036 (2025).

"Kendall transformation brings a robust categorical representation of ordinal data" M.B. Kursa. SciRep 12, 8341 (2022).

`write.dot` *Export tree to a Graphviz DOT format*

Description

Exports the vistla tree in a DOT format, which can be later layouted and rendered by Graphviz programs like dot or neato.

Usage

```
write.dot(
  x,
  con,
 vstyle = list(shape = function(x) ifelse(x$depth < 0, "egg", ifelse(x$leaf, "box",
    "ellipse")), label = function(x) sprintf("\"%s\"", x$name)),
 estyle = list(penwidth = function(x) sprintf("%0.3f", 0.5 + x$score/max(x$score) *
    2.5)),
  gstyle = list(overlap = "\"prism\"", splines = "true"),
  direction = c("none", "fromY", "intoY")
)
```

Arguments

x	vistla object.
con	connection; passed to writeLines. If missing, the DOT code is returned as a character vector.
vstyle	vertex attribute list — should be a named list of Graphviz attributes like shape or penwidth. For elements which are strings or numbers, the value is copied as is as an attribute value. For elements which functions, though, the function is called on a vistla_tree object and should return a vector of values.
estyle	edge attribute list, behaves exactly like vstyle. When functions are called, the Y-vertex is not present.
gstyle	graph attribute list. Functions are not supported here.
direction	when set to "none", graph is undirected, otherwise directed, for "fromY", root is a source, while for "intoY", a sink.

Value

For a missing con argument, a character vector with the graph in the DOT format, invisible NULL otherwise.

Note

Graphviz attribute values can be either strings, like "some vertex" in label, or atoms, like box for shape. When returning a string value, you must supply quotes, otherwise it will be included as an atom.

The default value of gstyle may invoke long layout calculations in Graphviz. Change to list() for a fast but less aesthetic layout.

The function does no validation whether provided attributes or values are correct.

References

"An open graph visualization system and its applications to software engineering" E.R. Gansner, S.C. North. Software: Practice and Experience 30:1203-1233 (2000).

Index