Package ‘phylopath’

October 14, 2022

Type Package
Title Perform Phylogenetic Path Analysis
Version 1.1.3
Maintainer Wouter van der Bijl <wouter@zoology.ubc.ca>
URL https://Ax3man.github.io/phylopath/
BugReports https://github.com/Ax3man/phylopath/issues
License GPL-3
LazyData TRUE
Depends R (>= 2.10)
Imports ape (>= 4.1), ggm (>= 2.3), ggplot2 (>= 3.0.0), ggraph (>= 1.0.0), igraph (>= 1.0.1), MuMIn (>= 1.15.6), parallel, pbapply (>= 1.3-2), phylolm (>= 2.5), purrr (>= 0.2.3), tibble
RoxygenNote 7.1.2
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr
NeedsCompilation no
Author Wouter van der Bijl [aut, cre]
Repository CRAN
Date/Publication 2021-10-04 19:00:02 UTC

R topics documented:

average ................................................................. 2
average_DAGs ................................................................ 3
best .............................................................................. 4
choice ........................................................................... 5
average

Extract and average the best supported models from a phylogenetic path analysis.

Description

Extract and average the best supported models from a phylogenetic path analysis.

Usage

average(phylopath, cut_off = 2, avg_method = "conditional", ...)

Arguments

phylopath An object of class phylopath.

cut_off The CICc cut-off used to select the best models. Use Inf to average over all models. Use the best() function to only use the top model, or choice() to select any single model.

avg_method Either "full" or "conditional". The methods differ in how they deal with averaging a path coefficient where the path is absent in some of the models. The full method sets the coefficient (and the variance) for the missing paths to zero, meaning paths that are missing in some models will shrink towards zero. The conditional method only averages over models where the path appears, making it more sensitive to small effects. Following von Hardenberg & Gonzalez-Voyer 2013, conditional averaging is set as the default. Also see MuMin::model.avg().

... Arguments to pass to phylolm::phylolm and phylolm::phyloglm. If you specified options in the original phylo_path call you don’t need to specify them again.
Value

An object of class fitted_DAG.

Examples

candidates <- define_model_set(
  A = NL ~ RS,
  B = RS ~ NL + BM,
  .common = c(LS ~ BM, DD ~ NL, NL ~ BM)
)
p <- phylo_path(candidates, rhino, rhino_tree)
summary(p)

# Models A and B have similar support, so we may decide to take
# their average.

avg_model <- average(p)
# Print the average model to see coefficients, se and ci:
avg_model

## Not run:
# Plot to show the weighted graph:
plot(avg_model)

# One can see that an averaged model is not necessarily a DAG itself.
# This model actually has a path in two directions.

# Note that coefficients that only occur in one of the models become much
# smaller when we use full averaging:

coef_plot(avg_model)
coef_plot(average(p, method = 'full'))

## End(Not run)

average_DAGs
Perform model averaging on a list of DAGs.

Description

Perform model averaging on a list of DAGs.

Usage

average_DAGs(
  fitted_DAGs,
  weights = rep(1, length(coef)),
  avg_method = "conditional",
)
Arguments

- **fitted_DAGs**: A list of fitted_DAG objects containing coefficients and standard errors, usually obtained by using `est_DAG()` on several DAGs.
- **weights**: A vector of associated model weights.
- **avg_method**: Either "full" or "conditional". The methods differ in how they deal with averaging a path coefficient where the path is absent in some of the models. The full method sets the coefficient (and the variance) for the missing paths to zero, meaning paths that are missing in some models will shrink towards zero. The conditional method only averages over models where the path appears, making it more sensitive to small effects. Following von Hardenberg & Gonzalez-Voyer 2013, conditional averaging is set as the default. Also see `MuMIn::model.avg()`.
- **...**: Additional arguments passed to `MuMIn::par.avg()`.
  
  For details on the error calculations, see `MuMIn::par.avg()`.

Value

An object of class `fitted_DAG`, including standard errors and confidence intervals.

Examples

```r
# Normally, I would advocate the use of the phylo_path and average
# functions, but this code shows how to average any set of models. Note
# that not many checks are implemented, so you may want to be careful and
# make sure the DAGs make sense and contain the same variables!
candidates <- define_model_set(
  A = NL ~ BM,
  B = NL ~ LS,
  .common = c(LS ~ BM, DD ~ NL)
)
fit_cand <- lapply(candidates, est_DAG, rhino, rhino_tree,
  model = \"lambda\", method = \"logistic_MPLE\")
ave_cand <- average_DAGs(fit_cand)
coef_plot(ave_cand)
```

**Description**

Extract and estimate the best supported model from a phylogenetic path analysis.
choice

Usage

best(phylopath, ...)

Arguments

phylopath An object of class phylopath.

... Arguments to pass to phylolm::phylolm and phylolm::phyloglm. If you specified options in the original phylo_path call you don’t need to specify them again.

Value

An object of class fitted_DAG.

Examples

candidates <- define_model_set(
  A = NL ~ BM,
  B = NL ~ LS,
  .common = c(LS ~ BM, DD ~ NL)
)
p <- phylo_path(candidates, rhino, rhino_tree)
best_model <- best(p)
# Print the best model to see coefficients, se and ci:
best_model
# Plot to show the weighted graph:
plot(best_model)

choice

Extract and estimate an arbitrary model from a phylogenetic path analysis.

Description

Extract and estimate an arbitrary model from a phylogenetic path analysis.

Usage

choice(phylopath, choice, ...)

Arguments

phylopath An object of class phylopath.

choice A character string of the name of the model to be chosen, or the index in model_set.

... Arguments to pass to phylolm::phylolm and phylolm::phyloglm. If you specified options in the original phylo_path call you don’t need to specify them again.
Value

An object of class fitted_DAG.

Examples

candidates <- define_model_set(
  A = NL ~ BM,
  B = NL ~ LS,
  .common = c(LS ~ BM, DD ~ NL)
)
p <- phylo_path(candidates, rhino, rhino_tree)
my_model <- choice(p, "B")
# Print the best model to see coefficients, se and ci:
my_model
# Plot to show the weighted graph:
plot(my_model)

cichlids

Cichlid traits and the evolution of cooperative breeding.

Description

A data set with binary traits, used in an analysis on the evolution of cooperative breeding by Dey et al 2017. Variable names are shortened for easy of use and consist of cooperative breeding (C), mating system (M), parental care (P), social grouping (G) and diet (D). All traits are coded as two level factors.

Usage

cichlids

Format

An object of class data.frame with 69 rows and 5 columns.

Source

**cichlids_tree**

**Cichlid phylogeny.**

**Description**

The phylogenetic tree of cichlid species that accompanies the cichlids dataset. The phylogeny is based on five nuclear genes and three mitochondrial genes.

**Usage**

```r
cichlids_tree
```

**Format**

An object of class `phylo` of length 4.

**Source**


---

**coef_plot**

**Plot path coefficients and their confidence intervals or standard errors.**

**Description**

Plot path coefficients and their confidence intervals or standard errors.

**Usage**

```r
ccoef_plot(  
fitted_DAG,
  error_bar = "ci",
  order_by = "default",
  from = NULL,
  to = NULL,
  reverse_order = FALSE
)
```
Arguments

- `fitted_DAG` A fitted DAG, usually obtained by `best()`, `average()` or `est_DAG()`.
- `error_bar` Whether to use confidence intervals ("ci") or standard errors ("se") as error bars. Will force standard errors with a message if confidence intervals are not available.
- `order_by` By "default", the paths are ordered as in the model that is supplied. Usually this is in the order that was established by `phylo_path()` for all combined graphs. This can be change to "causal" to do a reordering based on the model at hand, or to "strength" to order them by the standardized regression coefficient.
- `from` Only show path coefficients from these nodes. Supply as a character vector.
- `to` Only show path coefficients to these nodes. Supply as a character vector.
- `reverse_order` If TRUE, the paths are plotted in reverse order. Particularly useful in combination with `ggplot2::coord_flip()` to create horizontal versions of the plot.

Value

A ggplot object.

Examples

```r
D <- DAG(LS ~ BM, NL ~ BM, DD ~ NL + LS)
plot(D)
D_fitted <- est_DAG(D, rhino, rhino_tree, 'lambda')
plot(D_fitted)
coef_plot(D_fitted, error_bar = "se")
# to create a horizontal version, use this:
coef_plot(D_fitted, error_bar = "se", reverse_order = TRUE) + ggplot2::coord_flip()
```

Description

This function is a simple wrapper around the function from the ggm package with the same name. The only differences are that the `order` argument defaults to TRUE and that it adds a DAG class for easy plotting. Typically, one would use `define_model_set()` to create models for use with the phylopath package.

Usage

```r
DAG(..., order = TRUE)
```
Define a model set.

Arguments

... a sequence of model formulae

order logical, defaulting to TRUE. If TRUE the nodes of the DAG are permuted according to the topological order. If FALSE the nodes are in the order they first appear in the model formulae (from left to right). For use in the phylopath package, this should always be kept to TRUE, but the argument is available to avoid potential problems with masking the function from other packages.

Details

Supply a formulas for the model as arguments. Formulas should be of the form child ~ parent and describe each path in your model. Multiple children of a single parent can be combined into a single formula: child ~ parent1 + parent2. Finally, an isolate (unconnected variable) can be included as being connected to itself: isolate ~ isolate'.

Value

An object of classes matrix and DAG

Examples

# Use formula notation to create DAGs:
plot(DAG(A~B, B~C))
# Use + to easily add multiple parents to a node:
plot(DAG(A~B+C))
# Add a node as it's own parent to create an isolate:
plot(DAG(A~B+C, D~D))

Description

This is a convenience function to quickly and clearly define a set of causal models. Supply a list of formulas for each model, using either c(). Formulas should be of the form child ~ parent and describe each path in your model. Multiple children of a single parent can be combined into a single formula: child ~ parent1 + parent2.

Usage

define_model_set(..., .common = NULL)

Arguments

... Named arguments, which each are a lists of formulas defining the paths of a causal model.

.common A list of formulas that contain causal paths that are common to each model.
Details

This function uses `ggm::DAG()`.

Value

A list of models, each of class `matrix` and `DAG`.

Examples

```r
(m <- define_model_set(
  A = c(a~b, b~c),
  B = c(b~a, c~b),
  .common = c(d~a))
plot_model_set(m)
```

---

**est_DAG**  
Add standardized path coefficients to a DAG.

Description

Add standardized path coefficients to a DAG.

Usage

```r
est_DAG(DAG, data, tree, model, method, boot = 0, ...)
```

Arguments

- **DAG**: A directed acyclic graph, typically created with `DAG`.
- **data**: A `data.frame` with data. If you have binary variables, make sure they are either character values or factors!
- **tree**: A phylogenetic tree of class `phylo`.
- **model**: The evolutionary model used for the regressions on continuous variables. See `phylolm::phylolm` for options and details. Defaults to Pagel’s lambda model.
- **method**: The estimation method for the binary models. See `phylolm::phyloglm` for options and details. Defaults to logistic MPLE.
- **boot**: The number of bootstrap replicates used to estimate confidence intervals.
- **...**: Arguments passed on to `phylolm`: `lower.bound`: optional lower bound for the optimization of the phylogenetic model parameter.  
`upper.bound`: optional upper bound for the optimization of the phylogenetic model parameter.  
`starting.value`: optional starting value for the optimization of the phylogenetic model parameter.
measurment_error: a logical value indicating whether there is measurement error sigma2_error (see Details).
Arguments passed on to phyloglm:
btol: bound on the linear predictor to bound the searching space.
log.alpha.bound: bound for the log of the parameter alpha.
start.beta: starting values for beta coefficients.
start.alpha: starting values for alpha (phylogenetic correlation).

Value
An object of class fitted_DAG.

Examples
```
d <- DAG(LS ~ BM, NL ~ BM, DD ~ NL + LS)
plot(d)
d_fitted <- est_DAG(d, rhino, rhino_tree, 'lambda')
plot(d_fitted)
```

phylo_path: Compare causal models in a phylogenetic context.

Description
Continuous variables are modeled using phylolm::phylolm, while binary traits are modeled using phylolm::phyloglm.

Usage
```
phylo_path(
  model_set,
  data,
  tree,
  model = "lambda",
  method = "logistic_MPLE",
  order = NULL,
  parallel = NULL,
  na.rm = TRUE,
  ...
)
```

Arguments
```
model_set A list of directed acyclic graphs. These are matrices, typically created with define_model_set.
data A data.frame with data. If you have binary variables, make sure they are either character values or factors!
```
A phylogenetic tree of class phylo.

The evolutionary model used for the regressions on continuous variables. See `phylolm::phylolm` for options and details. Defaults to Pagel's lambda model

The estimation method for the binary models. See `phylolm::phyloglm` for options and details. Defaults to logistic MPLE.

Causal order of the included variable, given as a character vector. This is used to determine which variable should be the dependent in the dsep regression equations. If left unspecified, the order will be automatically determined. If the combination of all included models is itself a DAG, then the ordering of that full model is used. Otherwise, the most common ordering between each pair of variables is used to create a general ordering.

An optional vector containing the virtual connection process type for running the chains in parallel (such as "SOCK"). A cluster is create using the parallel package.

Should rows that contain missing values be dropped from the data as necessary (with a message)?

Arguments passed on to `phylolm`:
- `lower.bound`: optional lower bound for the optimization of the phylogenetic model parameter.
- `upper.bound`: optional upper bound for the optimization of the phylogenetic model parameter.
- `starting.value`: optional starting value for the optimization of the phylogenetic model parameter.
- `measurement_error`: a logical value indicating whether there is measurement error `sigma2_error` (see Details).

Arguments passed on to `phyloglm`:
- `btol`: bound on the linear predictor to bound the searching space.
- `log.alpha.bound`: bound for the log of the parameter `alpha`.
- `start.beta`: starting values for beta coefficients.
- `start.alpha`: starting values for `alpha` (phylogenetic correlation).

A phylopath object, with the following components:

- `d_sep` for each model a table with separation statements and statistics.
- `model_set` the DAGs
- `data` the supplied data
- `tree` the supplied tree
- `model` the employed model of evolution in `phylolm`
- `method` the employed method in `phyloglm`
- `dots` any additional arguments given, these are passed on to downstream functions
- `warnings` any warnings generated by the models
plot.DAG

Examples

# see vignette('intro_to_phylopath') for more details
candidates <- define_model_set(
  A = NL ~ BM,
  B = NL ~ LS,
  .common = c(LS ~ BM, DD ~ NL)
)
p <- phylo_path(candidates, rhino, rhino_tree)

# Printing p gives some general information:
p
# And the summary gives statistics to compare the models:
summary(p)

plot.DAG

Plot a directed acyclic graph.

Description

Plot a directed acyclic graph.

Usage

## S3 method for class 'DAG'
plot(
  x,
  labels = NULL,
  algorithm = "sugiyama",
  manual_layout = NULL,
  text_size = 6,
  box_x = 12,
  box_y = 8,
  edge_width = 1.5,
  curvature = 0.02,
  rotation = 0,
  flip_x = FALSE,
  flip_y = FALSE,
  arrow = grid::arrow(type = "closed", 18, grid::unit(15, "points")),
  ...
)

Arguments

x A ‘DAG’ object, usually created with the define_model_set() or DAG() function.
An optional set of labels to use for the nodes. This should be a named vector, of the form `c(var1 = "label1", var2 = "label2")`. If left at `NULL`, the variable names of the DAGs are used.

A layout algorithm from igraph, see `ggraph::create_layout()`. By default, uses the Sugiyama layout algorithm, which is designed to minimize edge crossing in DAGs.

Alternatively, precisely define the layout yourself, by providing a data.frame that at least has a column name with all variable names, and columns `x` and `y` with positions to be plotted. Setting this parameter overrides `algorithm` but other changes, such as rotation and flips will still be applied.

Size of the node label text.

To avoid the arrows colliding with the nodes, specify the rectangular dimensions of an invisible box around each node. If you have long labels, you need to increase this.

To avoid the arrows colliding with the nodes, specify the rectangular dimensions of an invisible box around each node. If you have multi-line labels, you need to increase this.

Width of the edges.

Curvature of the edges. A slight curvature can look pretty.

Supply the degrees you want to rotate the layout by. This is useful in order to put rotate your upstream nodes towards the top if needed.

Whether to flip the node positions horizontally.

Whether to flip the node positions vertically.

A `grid::arrow` object, specifying the shape and size of the arrowheads.

The order of facets is taken from the ordering of the list, with the facet labels coming from the names of the list. If the list is unnamed, sequential lettering is used.

Not used.

Examples

```r
    d <- DAG(a ~ b + c + d)
    plot(d)

    # Plot with manually defined positions:
    ml <- data.frame(
        name = c('a', 'b', 'c', 'd'),
        x = c(1, 1, 2, 2),
        y = c(1, 2, 1, 2)
    )
    plot(d, manual_layout = ml)
```
plot.fitted_DAG

Plot a directed acyclic graph with path coefficients.

Description

Plot a directed acyclic graph with path coefficients.

Usage

```r
## S3 method for class 'fitted_DAG'
plot(
  x,
  type = "width",
  labels = NULL,
  algorithm = "sugiyama",
  manual_layout = NULL,
  text_size = 6,
  box_x = 12,
  box_y = 8,
  edge_width = 1.25,
  curvature = 0.02,
  rotation = 0,
  flip_x = FALSE,
  flip_y = FALSE,
  arrow = grid::arrow(type = "closed", 18, grid::unit(15, "points")),
  colors = c("firebrick", "navy"),
  show.legend = TRUE,
  width_const = NULL,
  ...
)
```

Arguments

- **x**: An object of class fitted_DAG.
- **type**: How to express the weight of the path. Either "width", or "color".
- **labels**: An optional set of labels to use for the nodes. This should be a named vector, of the form c(var1 = "label1", var2 = "label2"). If left at 'NULL', the variable names of the DAGs are used.
- **algorithm**: A layout algorithm from igraph, see ggraph::create_layout(). By default, uses the Sugiyama layout algorithm, which is designed to minimize edge crossing in DAGs.
- **manual_layout**: Alternatively, precisely define the layout yourself, by providing a data.frame that at least has a column name with all variable names, and columns x and y with positions to be plotted. Setting this parameter overrides algorithm but other changes, such as rotation and flips will still be applied.
text_size  Size of the node label text.
box_x     To avoid the arrows colliding with the nodes, specify the rectangular dimensions of an invisible box around each node. If you have long labels, you need to increase this.
box_y     To avoid the arrows colliding with the nodes, specify the rectangular dimensions of an invisible box around each node. If you have multi-line labels, you need to increase this.
edge_width Width of the edges.
curvature Curvature of the edges. A slight curvature can look pretty.
rotation Supply the degrees you want to rotate the layout by. This is useful in order to put rotate your upstream nodes towards the top if needed.
flip_x    Whether to flip the node positions horizontally.
flip_y    Whether to flip the node positions vertically.
arrows    A grid::arrow object, specifying the shape and size of the arrowheads.
The order of facets is taken from the ordering of the list, with the facet labels coming from the names of the list. If the list is unnamed, sequential lettering is used.
colors    The end points of the continuous color scale. Keep in mind that red and green are obvious colors to use, but are better to be avoided because of color blind users.
show.legend Whether a legend for the color scale should be shown.
width_const Deprecated.
...       Not used.

Examples

d <- DAG(LS ~ BM, NL ~ BM, DD ~ NL + LS)
d_fitted <- est_DAG(d, rhino, rhino_tree, 'lambda')
plot(d_fitted)

---

**plot_model_set**  *Plot several causal hypothesis at once.*

**Description**

Plot several causal hypothesis at once.
Usage

```r
plot_model_set(
  model_set,
  labels = NULL,
  algorithm = "kk",
  manual_layout = NULL,
  text_size = 5,
  box_x = 12,
  box_y = 10,
  edge_width = 1,
  curvature = 0.05,
  rotation = 0,
  flip_x = FALSE,
  flip_y = FALSE,
  nrow = NULL,
  arrow = grid::arrow(type = "closed", 15, grid::unit(10, "points"))
)
```

Arguments

- **model_set** A list of DAG objects, usually created with `define_model_set()`.
- **labels** An optional set of labels to use for the nodes. This should be a named vector, of the form `c(var1 = "label1", var2 = "label2")`. If left at `NULL`, the variable names of the DAGs are used.
- **algorithm** A layout algorithm from igraph, see `ggraph::create_layout()`. By default, uses the Kamada-Kawai layout algorithm. Another good option is "sugiyama", which is designed to minimize edge crossing in DAGs. However, it can often plot nodes too close together.
- **manual_layout** Alternatively, precisely define the layout yourself, by providing a data.frame that at least has a column name with all variable names, and columns `x` and `y` with positions to be plotted. Setting this parameter overrides `algorithm` but other changes, such as `rotation` and `flips` will still be applied.
- **text_size** Size of the node label text.
- **box_x** To avoid the arrows colliding with the nodes, specify the rectangular dimensions of an invisible box around each node. If you have long labels, you need to increase this.
- **box_y** To avoid the arrows colliding with the nodes, specify the rectangular dimensions of an invisible box around each node. If you have multi-line labels, you need to increase this.
- **edge_width** Width of the edges.
- **curvature** Curvature of the edges. A slight curvature can look pretty.
- **rotation** Supply the degrees you want to rotate the layout by. This is useful in order to put rotate your upstream nodes towards the top if needed.
- **flip_x** Whether to flip the node positions horizontally.
- **flip_y** Whether to flip the node positions vertically.
**nrow**  
Number of rows to display the models on.

**arrow**  
A `grid::arrow` object, specifying the shape and size of the arrowheads.  
The order of facets is taken from the ordering of the list, with the facet labels  
coming from the names of the list. If the list is unnamed, sequential lettering is  
used.

**Value**  
A `ggplot` object.

**Examples**

```r
m <- list(one = DAG(a ~ b + c + d), two = DAG(a ~ b, b ~ c, d ~ d))
plot_model_set(m)
plot_model_set(m, algorithm = "sugiyama")
```

---

**red_list**  
*Data on brain size, life history and vulnerability to extinction*

**Description**  
A dataset with continuous variables affecting the conservation Status of mammalian species (the  
IUCN red list of threatened species).

**Usage**

`red_list`

**Format**

An object of class `data.frame` with 474 rows and 7 columns.

**Details**

It includes the following variables: brain size (Br), body size (B), gestation period (G), litter size (L),  
weaning age (W), population density (P) and vulnerability to extinction (Status).

**Source**

vulnerability to extinction in mammals. Evolution 70:1364-1375. doi: 10.1111/evo.12943
**red_list_tree**

*Description*

This is the accompanying phylogeny for the red_list data set. It is based on the updated mammalian supertree by Bininda-Emonds et al. 2007 & Fritz et al. 2009.

*Usage*

red_list_tree

*Format*

An object of class `phylo` of length 4.

*Source*


**rhino**

*Description*

A simulated dataset, as used by Gonzalez-Voyer and Von Hardenberg as an example, containing variables on body mass (BM), litter size (LS), nose length (NL), dispersal distance (DD) and range size (RS).

*Usage*

rhino

*Format*

An object of class `data.frame` with 100 rows and 6 columns.
**Source**


---

| rhino_tree | Rhinogrades phylogeny. |

---

**Description**

A phylogenetic tree for the 100 species of the rhino dataset.

**Usage**

`rhino_tree`

**Format**

An object of class `phylo` of length 4.

**Source**


---

| show_warnings | Print out warnings from a phylopath analysis. |

---

**Description**

Use this function after running `phylo_path()` to conveniently print any generated warnings to the screen. You can either provide no arguments, which will only work if you run it directly after the analysis, or you have to provide the `phylopath` object manually.

**Usage**

`show_warnings(phylopath = NULL)`

**Arguments**

- `phylopath` A phylopath object of which the warnings should be printed.
Index

* datasets
  
cichlids, 6
  cichlids_tree, 7
  red_list, 18
  red_list_tree, 19
  rhino, 19
  rhino_tree, 20

average, 2
average(), 8
average_DAGs, 3

best, 4
best(), 2, 8

choice, 5
choice(), 2

coefficient_plot, 7
cichlids, 6

cichlids_tree, 7

DAG, 8

DAG(), 13
define_model_set, 9
define_model_set(), 8, 13, 17

est_DAG, 10
est_DAG(), 4, 8

ggm::DAG(), 10
ggplot2::coord_flip(), 8
ggraph::create_layout(), 14, 15, 17

igraph, 14

MuMIn::model.avg(), 2, 4
MuMIn::par.avg(), 4

phylo_path, 2, 5, 11

phylolm::phyloglm, 2, 5, 10–12

phylolm::phylolm, 2, 5, 10–12

plot.DAG, 13
plot.fitted_DAG, 15
plot_model_set, 16

red_list, 18
red_list_tree, 19
rhino, 19
rhino_tree, 20

show_warnings, 20