Package 'phyreg'

July 23, 2025

Type Package

Title The Phylogenetic Regression of Grafen (1989)

Version 1.0.2

Date 2018-04-12

Author Alan Grafen

Maintainer Alan Grafen <alan.grafen@sjc.ox.ac.uk>

Description Provides general linear model facilities (single y-variable, multiple x-variables with arbitrary mixture of continuous and categorical and arbitrary interactions) for cross-species data. The method is, however, based on the nowadays rather uncommon situation in which uncertainty about a phylogeny is well represented by adopting a single polytomous tree. The theory is in A. Grafen (1989, Proc. R. Soc. B 326, 119-157) and aims to cope with both recognised phylogeny (closely related species tend to be similar) and unrecognised phylogeny (a polytomy usually indicates ignorance about the true sequence of binary splits).

License GPL-2 | GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-12 16:35:16 UTC

Contents

phyreg-package *The Phylogenetic Regression of Grafen (1989)*

Description

The Phylogenetic Regression provides general linear model facilities for cross-species analyses, including hypothesis testing and parameter estimation, based on the now rather uncommon situation in which the uncertainty about a phylogeny is well represented as a polytomous tree (see below for further discussion). It uses branch lengths to account for recognised phylogeny (which makes the errors of more closely related species more similar), and the single contrast approach to account for unrecognised phylogeny (that a polytomy usually represents ignorance about which exact binary tree is true, and so one higher node should contribute only one degree of freedom to the test). One dimension of flexibility in the branch lengths is fitted automatically.

Details

Package:	phyreg
Type:	Package
Version:	1.0.2
Date:	2018-04-12
License:	GPL-2 | GPL-3

You need a dataset of species data, and a phylogeny for those species that is in either (i) a series of taxonomic vectors, (ii) a "phylo" object, (iii) in newick format, or (iv) a single vector of the kind used internally in this package. For branch lengths, you can use the default "Figure 2" method of Grafen (1989), or specify the height of each node, or give a height to each level of the taxonomic vectors. Then you can test for H0 of one model against an HA of another, where the difference can be one or more x-variables, or interactions, or both, where the x-variables can be continuous or categorical. You may choose which species to include and exclude in each analysis. Missing data is handled automatically: you needn't do anything special, though a species is simply omitted from an analysis for which it has missing data in the response or in the control or test variables. It is assumed that missing data has the standard R value of NA.

In all linear models, the logic of each test involves controlling for some model terms while adding some test term(s). In simple cases such as ordinary multiple regression, the same analysis will give lots of tests and the user need only work out what the (often implicit) control and test terms are for each test. With the phylogenetic regression, only one test can be performed for a given analysis, and it is necessary to be explicit each time about the control terms and the test terms. This is because the single contrast taken across the daughters of one higher node depends upon the residuals in the control model.

At the time of the development of the theory, the best representation of the biologist's uncertain knowledge about a phylogeny was a polytomous tree where the polytomies represented uncertainty about the true order of splitting. By 2018, it has for some years been different. Now the common situation is to have a list of binary phylogenies with some kind of weighting as to how strongly each binary phylogeny is compatible with the data used to create the phylogeny. `phyreg()` is about

the first, historical, kind of uncertainty, and not about the more modern kind. Thus, it will only occasionally be found useful today. However, I like to have the sophisticated test available in a convenient form, thanks to the structure of R. For example, all the output and many inner workings can be made available as variables after an analysis.

Version 1.0.1 (appeared on CRAN 2018-04-08) made the package compatible with updated rules for having packages on CRAN but did not change the functionality. Version 1.0.2 (2018-04-12) made the package compatible with more of those updated rules that were drawn to my attention only after 1.0.1 had appeared on CRAN, and these changes necessitated dropping the capacity to retain default parameter values across sessions - sorry!

Full details and examples are given under `phyreg`

Author(s)

Alan Grafen, with portions copied as follows.

(1) `read.newick` (used internally only) copied from Liam Rewell (see `phyfromnewick` for a more detailed acknowledgment)

(2) the definition of `ginv` has been copied from MASS (package comes with current R downloads; book is W.N. Venables and B.D. Ripley (2002) Modern Applied Statistics in S. (Fourth Edition), Springer – `http://www.stats.ox.ac.uk/pub/MASS4`). This avoids having to `require` the whole package, though it may mean I have amend it if R and MASS make a simultaneous change in the low-level routines they use. `ginv` was copied and pasted from R 3.0.2 on 64-bit MacOS on 24th January 2014. It finds the generalised inverse of a matrix.

(3) code for dealing with model formulae (internal functions `merge.formulae.ag` and `merge.formulae.test.ag`) was adapted from code of Steven Carlisle Walker obtained from `https://stevencarlislewalker.wordpress.com/2012/08/06/merging-combining-adding-together-two-formula-objects-in-r/` in January 2014.

Maintainer: Alan Grafen <alan.grafen@sjc.ox.ac.uk>

References

Grafen, A. 1989. The phylogenetic regression. Philosophical Transactions of the Royal Society B, 326, 119-157. Available online at `http://users.ox.ac.uk/~grafen/cv/phyreg.pdf`. Some further information including GLIM and SAS implementations is available at `http://users.ox.ac.uk/~grafen/phylo/index.html`.

See Also

`phyreg`

`Example datasets` *Example data*

Description

Datasets myd0 to myd3 are data frames that contain a y-variable, three continuous x-variables X1 to X3, and two factors A and B. They were simulated by a phylogenetic method, and also contain the error. The phylogeny was created from the taxonomic variables in extax – the heights of the levels were assumed to be 0 for species, 1,2,3 for the three taxonomic variables, and 4 for the root.

Usage

```
data(myd0)
data(myd1)
data(myd2)
data(myd3)
data(extax)
data(newickstr)
```

Format

myd0 to myd3 are datasets with 100 observations on the following 7 variables.

`error` drawn from a standard Normal distribution for each species, but of course correlated appropriately for the phylogeny

`X1-X3` continuous variables

`A, B` factors with four levels 1 to 4

`y` the response variable.

extax is a data frame containing three taxonomic variables describing a phylogeny for 100 species newickstr is a character variable containing the same phylogeny as represented by extax, but in newick format – see Description for explanation of the heights.

Details

The formula calculating y from the independent variables was the same in each dataset, but the independent variables and the error were resampled for each dataset. The mean square error increases from myd0 to myd3. By including "error" in the model (never possible in life, of course) you can obtain the coefficients and root mean square error used to construct the data. The categorical characters are evolved on the tree by taking a transition matrix (tm) to represent the transition probabilities after one unit of time, taking the matrix logarithm (ltm<-logm(tm)), and using the transition matrix expm(bl * ltm) for a branch length of duration bl. (Note that expm() and especially logm() are available in the package expm.)

For examples of use, see phyreg.

See Also

The data is useful in calls to phyreg, with arguments specified as data=myd0 etc and taxmatrix=extax. newickstr is useful in calls to phyfromnewick.

Examples

```
data(myd0, myd1, myd2, myd3, extax, newickstr)
```

factory_default *Default settings for "minor" parameters*

Description

These functions change the effect of including `reset=TRUE` as an argument to `phyreg()`. After a call of `factory_default()`, the reset will be to the original values hard-wired in the package's code. If you call `new_default()`, the values of the minor parameters in place at that moment become the reset values for all uses of `reset=TRUE` until a future call of one of these functions.

Usage

```
factory_default()
new_default()
```

Details

The unstored, major, parameters are the `control` and `test` parameters, and the data parameters, namely `data`, `phydata`, `taxmatrix` and `heightsdata`. The remaining parameters are minor. All the parameters are listed under phyreg, and examples of use are also given there. Note that changing a parameter whether major or minor leaves the newly set value of that parameter as the default for future calls of `phyreg()` until either the parameter is changed again, or `reset=TRUE` is included as an argument of `phyreg()`.

Author(s)

Alan Grafen

See Also

phyreg

inf *Provides information about the stored output of a call to phyreg()*

Description

After an instruction `m<-phyreg(...)`, you can type `inf(m,...)` to provide information about `m`. You can see output again, or that you didn't ask for from the initial call of `phyreg()`. And you can find out whether optional storage took place of the different possible outputs, if not, how to ask for it, and if so, how to access it.

Usage

```
inf(inpr, ..., oppf = 5, opdf = 1, oprho = 0, dfwarning = 1)
```

Arguments

inpr	An object of class phyreglm i.e. that has been produced from an assignment of the form m<-phyreg(...).
...	To see the list of the possible parameters, call inf() itself with no parameters. They are arguments of phyreg that influence printing and storage of information. Four are shown below and can vary the way information is presented compared to the original call. See Details for information on the others.
oppf	If non-zero, the p-value and F-statistic are printed out at each call of phyreg, with ndigits=oppf. If zero, they are not printed out. (inf) will print these out for you if you include "oppf".
opdf	If non-zero, a breakdown of degrees of freedom is printed out at each call of phyreg. (inf) will print this out for you if you include "opdf".
oprho	If nonzero, details of rho will be printed with ndigits=oprho. If rho was fitted, the value of rho and the log-likelihood will be given, and how the search ended (lower boundary, or an internal maximum). (inf) will print this out for you if you include "oprho".
dfwarning	If TRUE or 1, the loss of degrees of freedom for phylogenetic reasons is detailed. (inf) will print this out for you if you include "dfwarning".

Details

The ... are used rather irregularly to allow values to be specified as character constants. These are the 13 final arguments of phyreg(), which specify material to be printed or to be stored. The printed material will be reprinted by inf(), saving the need to repeat the analysis. For saved material, including the character constant will instruct inf() to tell you whether that material was stored and both how to access it if it was stored and , if not, how to store it next time you use (phyreg). You may include as many as these character constants as you like in one call of inf(). The "printing" constants are "parmx", "parmxz", "opfunccall", "means", and "addDF", while the "saving" constants are "linputs", "sinputs", "lmshortx", "lmshortxz", "lmlongx", "lmlongxz", "hinput" and "paper". To find what information is stored or printed by each character constant, see the help file for phyreg and look for the argument with the same name.

Author(s)

Alan Grafen

See Also

phyreg

phyreg *Performs a phylogenetic regression*

Description

If this documentation looks daunting, look first at the examples below!

Takes a control formula, some test terms, a dataframe, a phylogeny, and optionally a set of node heights, and performs a test of H0 (control terms only) vs HA (control plus test terms). The user can specify a value of rho (a parameter that determines whether branch lengths are bigger near the root or near the tips), or ask that it be fitted by maximum likelihood. The primary outputs are a p-value and F-ratio, and parameter estimates. Much more information is available, much of it needing to be interpreted very cautiously.

Usage

```
phyreg(control, test, data, subset, phydata, taxmatrix, heightsdata,
rho = -1, lorho = 0.3, hirho = 0.6, errrho = 0.02, minrho = 1e-04,
tolerance = 1e-06, oppf = 5, opdf = 0, parmx = 0, parmxz = 0,
opfunccall = 0, addDF = 0, linputs = FALSE, sinputs = FALSE, means = FALSE,
lmshortx = FALSE, lmshortxz = FALSE, lmlongx = FALSE, lmlongxz = FALSE,
hinput = FALSE, paper = FALSE, dfwarning = TRUE, oprho = FALSE, reset = FALSE)
```

Arguments

control	The null hypothesis model, which should be a formula with a response variable. If there are no variables to control for, specify y~1. (y~0 is not permitted, as it undermines the logic of the test.)
test	The test terms. To control for y~b, and test for a*x, you can specify (i) test="a*x" or (ii) test=y~b+a*x or (iii) test=~+a*x. Note the "+" *and* the absence of the response variable in the final form
data	A mandatory dataframe containing the variables in the model formulae
subset	a vector of the same length as the data, containing 0/1 for exclusion or inclusion of a species. If unspecified, the internal default is to include all species. To return to being unspecified, say subset=NULL
phydata	a vector with the phylogeny in the internal package format (containing for each non-root node the ID number of its parent). Either phydata or taxmatrix must be supplied. To unspecify, say phydata=NULL or taxmatrix=NULL. A phylogeny in newick or phylo format can be converted into the internal forma, preserving height information, with phyfromnewick or phyfromphylo.
taxmatrix	a dataframe with a set of taxonomic vectors to specify the working phylogeny. There should be no species column, and the others should start with low levels (e.g. genus) and end with the highest (perhaps order). If taxmatrix is specified, you can alternatively specify heightsdata with a vector containing one height for each level you've given, plus a height for the root, or if that is NULL the default "Figure 2" of Grafen (1989) will be used. Either taxmatrix or phydata (– not both) must be supplied. Despecify with taxmatrix=NULL.

`heightsdata` If not specified, the height of nodes will be calculated using the default "Figure 2" method of Grafen (1989). If a height is specified for each node, then the vector will be one element longer than `phydata`, as the root has a height but no parent. If `taxmatrix` is specified, then `heightsdata` can also be specified as a vector of length one plus the number of taxonomic vectors, in which case each node is given a height corresponding to its level in the taxonomy – the extra element is the height of the root. In both cases, `heightsdata` must be non-decreasing.

`rho` If zero or negative, `rho` will be fitted by maximum likelihood. If positive, that value will taken for rho and no fitting will be done. Once the node heights are scaled to lie between 0 and 1, each height is raised to the power rho before being used to determine the error structure in the model. See Grafen (1989) for details.

`lorho` The starting lower bound for the search for rho

`hirho` The starting upper bound for the search for rho

`errrho` The search for rho will stop when it is known to lie in an interval of length `errrho`

`minrho` The search for rho will stop when the whole current search interval lies below `minrho`. A zero value of rho is problematic for the program.

`tolerance` This tolerance decides whether there is enough variability among the daughters of a higher node to justify putting it into the short regression. This is because the residuals in the long regression on the control variables are used to calculate a contrast; if they are very small, then the contrast is effectively being determined by machine rounding errors, and if they are zero, then all contrasts should also be zero and so the higher node will be irrelevant in a regression through the origin.

`oppf` If non-zero, the p-value and F-statistic are printed out at each call of `phyreg`, with `ndigits=oppf`. If zero, they are not printed out.

`opdf` If non-zero, a breakdown of degrees of freedom is printed out at each call of `phyreg`.

`parmx` If non-zero, the parameters in the long regression with the control formula are printed out at each call of `phyreg`.

`parmxz` If non-zero, the parameters in the long regression with the control+test formula are printed out at each call of `phyreg`. Note these should be regarded with caution, as the value of rho was fitted just on the control formula. For better estimates, run `phyreg` again with the old control+test as the new control, and set `opparmx=1`. But `opparmxz` will usually give a good rough guide.

`opfunccall` If non-zero, the function call is printed out

`addDF` Advised to leave well alone. Included for backwards compatibility with the original GLIM version. It was originally but misguidedly intended to make a correction to the total degrees of freedom for the test in recognition of the fitting of rho. However, the fitting of rho is now regarded as affecting the numerator and denominator SS equally, and so not to require the correction.

`linputs` If `TRUE` or `1`, the inputs to the long regression will be stored. After `x<-phyreg(...,linputs=1)`, it will be found that `x$linputs$y`, `x$linputs$designxz` and `x$linputs$w` will contain the response, design matrix, and weights for the long regression,

respectively. Use with care. The whole point of the the problem of *unrecognised* phylogeny is that the standard errors from this regression cannot be trusted, though with a dependable binary phylogeny, the output can be accepted at face value.

`sinputs` If `TRUE` or `1`, the inputs to the short regression will be stored. After `x<-phyreg(...,sinputs=1)`, it will be found that `x$sinputs$y`, `x$sinputs$designxz` and `x$sinputs$w` will contain the response, design matrix, and weights for the short regression, respectively. Use with care. Not only can the standard errors from this regression not be trusted, but the parameter estimates are biassed!

`means` If `TRUE` or `1`, the phylogenetically-weighted means will be printed for the response and for each variable in the design matrix of the control+test model (in the original species dataset, which are the same as the weighted means in the long regression).

`lmshortx` If `TRUE` or `1`, then after `x<-phyreg(...,lmshortx=1)`, the `lm.wfit()` output from the short regression with the control formula will be stored in `x$lmshortx`. Use with care – the parameter estimates are biassed...

`lmshortxz` If `TRUE` or `1`, then after `x<-phyreg(...,lmshortxz=1)`, the `lm.wfit()` output from the short regression with the control+test formula will be stored in `x$lmshortxz`. Use with care – the parameter estimates are biassed...

`lmlongx` If `TRUE` or `1`, then after `x<-phyreg(...,lmlongx=1)`, the `lm.wfit()` output from the long regression with the control formula will be stored in `x$lmlongx`. The SEs cannot be trusted for non-binary phylogenies.

`lmlongxz` If `TRUE` or `1`, then after `x<-phyreg(...,lmlongxz=1)`, the `lm.wfit()` output from the long regression with the control+test formula will be stored in `x$lmlongxz`. The SEs cannot be trusted for non-binary phylogenies.

`hinput` If `TRUE` or `1`, then after `x<-phyreg(...,hinput=1)`, the heights used will be stored in `x$hinput`. If you use `taxmatrix=...`, this is a way to obtain the internal "phy" version of the heights for each higher node. These heights have *not* been modulated by rho i.e. they are the starting heights before rho is applied.

`paper` If `TRUE` or `1`, then after `x<-phyreg(...,paper=1)`, various matrices that appear in the appendix of the source paper (Grafen 1989) will be stored under x$paper. Type `names(x$paper)` to see which ones!

`dfwarning` If `TRUE` or `1`, the loss of degrees of freedom for phylogenetic reasons is detailed.

`oprho` If nonzero, details of rho will be printed with `ndigits=oprho`. If rho was fitted, the value of rho and the log-likelihood will be given, and how the search ended (lower boundary, or an internal maximum).

`reset` If `TRUE` or `1`, the minor parameters will all be reset to their default values. Useful if you've forgotten what you've set, or if you're moving on to another analysis. See `new_default` and `factory_default` for changing the default values themselves, and for what a *minor* parameter is.

Details

One key point for using the program is that each node in the phylogeny has an ID number, with which it is identified. For a species, the number is given by its position in the data frame with

the data itself, which is assumed to have species in the same order as in the taxonomic variables data frame, if that is specified, and the same order as in the newick-style phylogeny if you use `phyfromnewick`. If `phyfromphylo` is used, the code numbers of the species are carried over from the "phylo" object to the internally formatted "phy" vector. These species ID numbers are fixed no matter whether that species or another is excluded by the `subset` parameter of `phyreg`, or for missing data. Higher nodes have ID numbers too. In every phylogeny, each node's parent has a higher ID number than it does, except the root, which has no parent. The numbers for higher nodes *do* vary when species are omitted for any reason.

The main feature of phylogenetic data that the Phylogenetic Regression deals with differently from other methods is *unrecognised phylogeny*. It operates on the principle that each higher node should provide one datapoint to a valid test, and not more, and does so by choosing just one linear contrast across the daughters of each higher node. The contrast coefficients come from the residuals of the long regression on the control model. For a full mathematical justification of this choice, see the appendix to Grafen (1989), and see the paper itself for conceptual explanations. With binary phylogenies, of course, there is no unrecognised phylogeny. A parameter of the branch lengths is fitted automatically, unless the user wishes to impose a value, which allows the strength of phylogenetic association to be make weaker or stronger. Simulation studies in Grafen (1989) show that the method has good properties, and also that ignoring unrecognised phylogeny can create serious problems.

On some occasions, degrees of freedom are lost "for phylogenetic reasons". A whole node may be lost to the final test if the residuals of its daughter nodes are all zero in the long regression. This can happen for various reasons, most often when (i) the response is in fact binary, and so there is no variation in it below a node, or (ii) a categorical variable has so many values restricted to one part of the tree that a subset of its parameter values can adjust to render all the residuals zero in that part of the tree. That is called a node being lost in the denominator. The other possibility is that once the contrasts have been taken across each higher node, the design matrix for the model has lower rank than it did before, which is called losing a degree of freedom in the numerator (it is transferred to the denominator). You can choose to be warned when degrees of freedom are lost for phylogenetic reasons (use `dfwarning=1`), or to see a whole breakdown of degrees of freedom including any lost (use `opdf=1`). If nodes are lost in the denominator, their ID numbers are stored in the output `$shornode`, though note this also contains an initial `0` for programming convenience. These phylogenetic degree of freedom issues need to be handled properly to provide a valid test – see Grafen (1989) for details.

The data for long and short regressions, and the `lm.wfit()` output are provided on request, but must be used with great caution. The whole point of the phylogenetic regression is that neither type of regression provides results that can be taken at face value, except the long regression under binary phylogenies. Use at your own risk! See Grafen (1989) for details.

Some simulated data is included to facilitate the examples below – see `myd0`

Value

`H0model`	The control model
`HAmodel`	The control+test model
`spu`	A vector with a 0/1 for each species to indicate whether it was used (1) or not (0), depending on the argument `subset` and on missing values
`nomspuse`	The number of species omitted because of missing values (not counting those already omitted because of the argument `subset`)

`longrss` The residual sum of squares in the long regression with the control formula

`missingnodes` A vector of the ID numbers of nodes omitted for phyogenetic reasons (see Details above). An extra zero is included for programming convenience.

`shornode` The ID numbers of the higher nodes in the supplied phylogeny that are used in the short regression, so `shornode[5]` gives the ID number of the node for the 5th datapoint.

`details` contains various numbers needed for output, namely `$poff` (the p-value), `$testf` (the F-ratio), `$testnumdf` (the numerator DF), `$testdendf` (the denominator DF), `$nspec` (total number of species, included and excluded, so it equals nrow(input_dataframe)), `$nommiss` (the number of species omitted because of missing values (not counting those already omitted because of the argument `subset`)), `$nspecactive` (number of species included in the analysis), `$nomspuse` (number of species omitted because of the argument `subset`), `$nphyhinodes` (the number of non-species nodes in the supplied phylogeny (ie. with all species included)), `$hilostomsp` (the number of higher nodes lost through species omitted because of the argument `subset`, `$hilostvar` (number of higher nodes lost to the short regression through lack of variability in the long regression – see "Phylogenetic Degrees of Freedom" in Details), `$shortotdf` (total DF in short regression), `$dflx` (rank of the long regression with the control model), `$dfxlost` (loss of numerator DF in short regression – see "Phylogenetic Degrees of Freedom" in Details), `$dfsx` (rank of short regression with control model), `$testlongdf` (=`$dflxz-$dflx`, the degrees of freedom for the test terms in the long regression), `$testlost` (degrees of freedom for the test variables, lost for phylogenetic reasons – see "Phylogenetic Degrees of Freedom" in Details)), `$shortcondf` (control degrees of freedom in the short regression), `$addDF` (retains the argument `addDF` as it's needed in calculations), `$rho` (the value of rho used in the final test), `$lik` (the log-likelihood of that value of rho), `$edge` (coded -1 for error, 2 for rho set by the user, and, with a fitted rho, 0 for an internal maximum of the likelihood, and 1 for a maximum at the lower edge of the search region (specified by `minrho`)), `$missingnodes` (the numbers of the higher nodes omitted for lack of variability in the short regression, but an extra 0 at the start – see Details)

`means` The phylogenetically weighted means of the response variable and the design matrix for the control+test model

`parmx` Parameter estimates from the long regression with the control model

`parmxz` Parameter estimates from the long regression with the control+test model

`funccall` The function call with which you invoked `phyreg`

`fullphy` The full phylogeny for all species. This will just be `phydata` if you specified the phylogeny that way, but otherwise has been converted from taxonomic vectors.

`usedphy` The phylogeny for the species included in the analysis. There is an entry for every species, and each included species has its original ID, and every omitted species has zero. There may well be fewer higher nodes in `usedphy` than in `fullphy`, because only higher nodes with two or more daughters are retained. This vector is useful if you want to study the matrices from the paper (see the argument and `valuepaper`) as many are indexed by `usedphy`. All node numbers reported by the program elsewhere are the original IDs, and so `usedphy` is important only for internal and/or technical reasons.

originalIDs	This tells you, for each node in usedphy, which node in fullphy it corresponds to.
rho	A list with $rho, $lik and $edge – see $details for details
sinputs	(optional) the inputs to the short regression namely sinputs$y, sinputs$design and sinputs$w for the response, design matrix and weights
linputs	(optional) the inputs to the long regression namely linputs$y, linputs$design and linputs$w for the response, design matrix and weights
lmlongx	(optional) The lm.wfit() output from the long regression with the control model. Choose to store using the argument of the same name.
lmlongxz	(optional) The lm.wfit() output from the long regression with the control+test model. Choose to store using the argument of the same name.
lmshortx	(optional) The lm.wfit() output from the short regression with the control model. Choose to store using the argument of the same name.
lmshortxz	(optional) The lm.wfit() output from the short regression with the control+test model. Choose to store using the argument of the same name.

Author(s)

Alan Grafen, with portions copied as follows.

(1) read.newick (used internally only) copied from Liam Rewell (see phyfromnewick for a more detailed acknowledgment)

(2) the definition of ginv has been copied from MASS (package comes with current R downloads; book is W.N. Venables and B.D. Ripley (2002) Modern Applied Statistics in S. (Fourth Edition), Springer – http://www.stats.ox.ac.uk/pub/MASS4). This avoids having to require the whole package, though it may mean I have amend it if R and MASS make a simultaneous change in the low-level routines they use. ginv was copied and pasted from R 3.0.2 on 64-bit MacOS on 24th January 2014. It finds the generalised inverse of a matrix.

(3) code for dealing with model formulae (internal functions merge.formulae.ag and merge.formulae.test.ag) was adapted from code of Steven Carlisle Walker obtained from https://stevencarlislewalker.wordpress.com/2012/08/06/merging-combining-adding-together-two-formula-objects-in-r/ in January 2014.

References

The source paper is Grafen, A. 1989. The phylogenetic regression. Philosophical Transactions of the Royal Society B, 326, 119-157. Some further information is at http://users.ox.ac.uk/~grafen/phylo/index.html, including previous implementations in GLIM and SAS.

See Also

The package helpfile is at phyreg-package. Functions are inf, phyfromnewick, phyfromphylo, factory_default, new_default. For simulated data see myd0.

Examples

```
## First get some data
data(myd0,myd1, myd2, myd3, extax)
## Then do our first analysis
phyreg(y~X1,"A",myd0,taxmatrix=extax)
## and test instead for "B", noticing that only the changed parameter need be given
phyreg(test="B")
## and we do more complicated analysis involving an interaction with an existing term
phyreg(y~X1+X2,"A*X1")
## Now we choose to see the output relating to rho and to how the degrees of freedom are determined,
##   and we also wish to see the means for each variable, and the parameters from the long
##   regression on control+test variables
phyreg(oprho=6, opdf=1, means=1,parmxz=1)
## To illustrate inf, we store the results of an analysis in m1
m1<-phyreg(y~A,"X1+X2")
## Note we still get the extra output from the previous call, because those parameters
##   too are remembered within a session. But we can see it again, whether or not we
##   saw it the first time, with inf. inf reminds you if you forget quite how to use it
inf()
inf(m1)
inf(m1,oppf=3)
inf(m1,oppf=7, oprho=5)
inf(m1, oppf=5, "means", "parmx")
inf(m1,"sinputs","lmshortx")
## The final call asks for things m1 doesn't have because it wasn't stored at the time. Now
##  we turn to changing the default parameters with new_default and factory_default. The help
##  pages for those functions explain that only minor parameters (those affecting output and
##  storage, and those affecting rho), and not the models or the data-bearing parameters, have
##  default values.
new_default()
## This call takes the most recent parameters to phyreg and makes them the default, which
##   in the first instance changes nothing. But if we later call reset=TRUE as an argument
##   of phyreg, it is the values at the time of calling new_default() that will be returned
##   to. In this instance, we would automatically get output on rho and degrees
##   of freedom, and the means and xz parameters printed).
##
##   To see this in operation, we would change those parameters...
phyreg(oprho=0,opdf=0,means=0,parmxz=0)
##  ... and then in the next call use reset=1. That will restore the defaults, which we changed
##   with new_default. To change the defaults back to the values that shipped with the
##   package,
factory_default()
##   does the job.
##
## Finally, the phylogeny has so far been determined by a data frame of taxonomic variables in the
##  argument taxmatrix. If we have the phylogeny available in newick style, we can convert to
##  the internal format, and then use that instead. Fortunately, one is provided. Note it is
##  good form to unset the other method of specifying a phylogeny (which is being remembered by
##   the package) with taxmatrix=NULL
data(newickstr)
z<-phyfromnewick(text=newickstr)
phyreg(phydata=z$phy,taxmatrix=NULL)
```

```
##   ... and if branch lengths were supplied, and we trust them, we can
phyreg(y~X1, "A", phydata=z$phy, heightsdata=z$hts)
##  Similarly with a phylogeny in phylo format. We obtain one of those by using the "phylo"
##    value from phyfromnewick, and use an intermediate variable"phyloobject" to show how
##    to use one if you have one
phyloobject<-z$orphylo
phyconverted<-phyfromphylo(phyloobject)
phyreg(phydata=phyconverted$phy)
phyreg(phydata=phyconverted$phy, heightsdata=phyconverted$hts)
## The results should all be the same because it's the same phylogeny represented in three
##   different ways, with the same heights.
##
## Enjoy!
```

`Translating phylogenies`

Using phylogenies formatted in standard but "other" ways

Description

The two functions convert phylogenies in a standard format (newick or phylo) into a form that can be supplied as the argument `phydata` of `phyreg`. If node heights are specified, these are also provided in the value returned.

Usage

```
phyfromnewick(file = "", text)
phyfromphylo(phylo)
```

Arguments

`file`	a textfile containing the newick phylogeny, which must end with a semi-colon (`phyfromnewick` only)
`text`	a character variable containing the newick phylogeny, which must end with a semi-colon. Exactly one of these arguments should be supplied (`phyfromnewick` only)
`phylo`	an R-variable containing a "phylo" object, as used, for example, in the ape package (`phyfromphylo` only).

Details

Before passing the text to `read.newick`, I strip out all the spaces, to avoid problems.

`phyfromnewick` contains a function `read.newick` which "reads Newick style tree with branch lengths into memory as an ape "phylo" object", and was written by Liam J. Revell in 2011, and downloaded on 24th January 2014. See references for url.

Examples are given under `phyreg`.

Value

`phy`	The phylogenetic vector in internal format, with an element for every node, except the root, but including species. If `jj==phy[[ii]]`, then `jj` is the parent-node of `ii`. It is a requirement for the internal format that `jj>ii` in every case (it is *not* true of "phylo" objects that a parent's ID has to be greater than its offspring's). Suitable for supplying as the `phydata` argument of `phyreg`
`hts`	The heights of each node, a vector with an element for every node. Thus it is longer by one than $phy. Suitable for supplying as the `heightsdata` argument of `phyreg`
`orphylo`	The "phylo" style object created by `read.newick`. (`phylofromnewick` only)

Author(s)

Alan Grafen, with an internal function (`read.newick`) written by Liam J. Revell in 2011, downloaded on 24th January 2014. See references for url.

References

https://github.com/liamrevell/phytools/blob/master/R/read.newick.R

See Also

`phyreg`

Index