The caper package: comparative analysis of phylogenetics and evolution in R

David Orme

October 16, 2024

This vignette documents the use of the caper package for R (R Development Core Team, 2011) in carrying out a range of comparative analysis methods for phylogenetic data. The caper package, and the code in this vignette, requires the ape package (Paradis et al., 2004) along with the packages mvtnorm and MASS.

Contents

1 Background

Comparing the traits of species (or of groups of species of higher taxonomic rank) can produce deep insights into evolutionary processes. However, all such analyses should take into account the degree to which species are related and hence do not provide independent data on a hypothesis. This can lead both to situations in which apparently strong relationships rest on relatively few truly independent events (Fig. 1a) or where strong relationships within groups are masked by phylogenetic differences between groups (Fig. 1b).

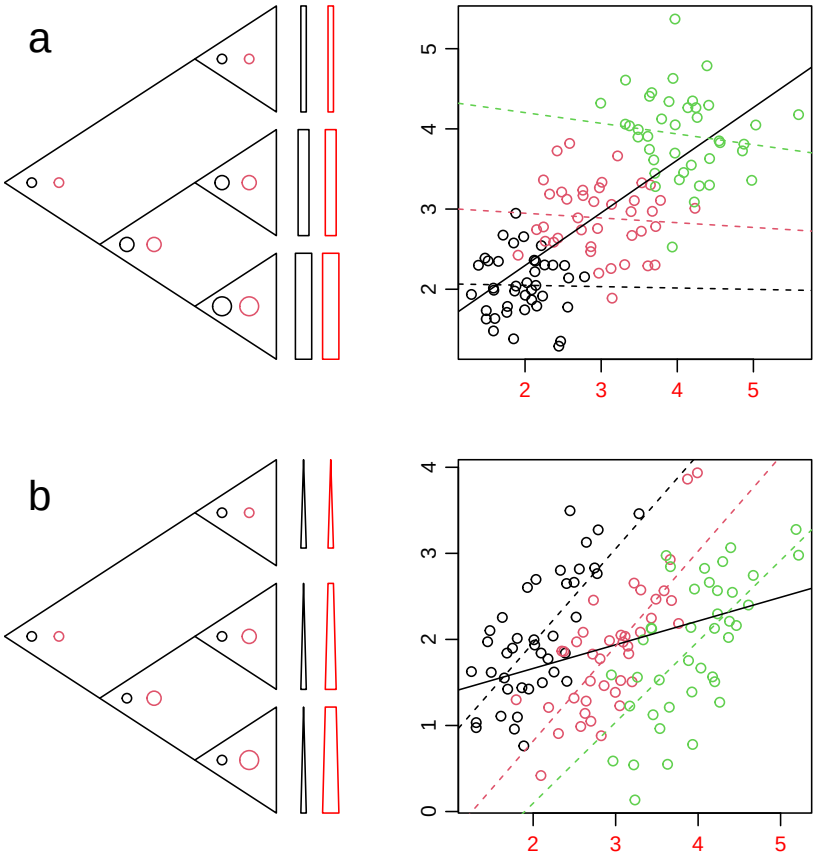


Figure 1: Phylogenetic autocorrelation in action. a) Simple regression (solid line) suggests a strong relationship between two variables; the phylogeny shows that within the three main groups, there is no consistent relationship (dashed lines) and the apparent relationship stems from only two linked shifts in the means of the traits early in the evolution of the clade. b) Simple regression (solid line) suggests a weak positive relationship between the two variables; the phylogeny shows that there are strong positive relationships between the traits within each of the three main groups but this is masked by early shifts in the mean value of the red trait.

The `caper` package implements two methods that account for this phylogenetic autocorrelation. The first, originally described by Felsenstein (1985), is to recognize that the differences between taxa on either side of a bifurcating node represent independent evolutionary trajectories and that these differences ('independent contrasts') can be used to test hypotheses in a way that accounts for the phylogenetic autocorrelation between the taxa. Pagel (1992) extended this method to permit contrasts to be calculated at polytomies. The second, described by [refs], is to include the phylogenetic structure as a covariance matrix in a linear model.

The `caper` package also provides a number of other phylogenetic comparative methods, along with a flexible method for simulating trait evolution on phylogenies.

2 Comparative datasets

2.1 The `comparative.data` class and objects.

The majority of functions in `caper` require both a phylogeny and data set of traits for the taxa at the tips of that phylogeny. Crucially, the rows of data need to be matched carefully to the tips to ensure that the phylogenetic structure in the variables is represented correctly: `caper` provides the function `comparative.data` to facilitate this. The following simple example shows the basic operation:

```
> phy <- read.tree(text='(((B:2,A:2):1,D:3):1,(C:1,E:1):3);')
> dat <- data.frame(
+   taxa=c("A","B","C","D","E"),
+   n.species=c(5,9,12,1,13), mass=c(4.1,4.5,5.9,3.0,6.0)
+ )
> cdat <- comparative.data(data=dat, phy=phy, names.col="taxa")
> print(cdat)
```

```
Comparative dataset of 5 taxa:
Phylogeny: phy
   5 tips, 4 internal nodes
   chr [1:5] "B" "A" "D" "C" "E"
Data: dat
   $ n.species: num [1:5] 9 5 1 12 13
   $ mass     : num [1:5] 4.5 4.1 3 5.9 6
```

In order to make it easier to use contrast methods, where values are estimated at internal nodes, the `comparative.data` function and contrast methods use internal node labels to identify values. Creating a comparative data object will not replace existing labels but will insert simple numeric labels for unlabelled nodes. The `comparative.data` class has a number of generic methods that allow users to extract a subset of taxa and the relevant linked data from within a `comparative.data` object.

2.1.1 `na.omit`

This method returns a subset of the object containing only those taxa for which the data are complete (i.e no `NA` values). By default, the `comparative.data` function drops incomplete rows, but this can be altered. The examples below show the function in use on the `perissodactyla` dataset.

```
> data(perissodactyla)
> (perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial))
```

```
Comparative dataset of 9 taxa:
Phylogeny: perissodactyla.tree
   9 tips, 7 internal nodes
   chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:9] 6.24 6.18 6.26 5.46 5.65 ...
   $ log.gestation.length: num [1:9] 2.68 2.65 2.69 2.6 2.59 ...
   $ log.neonatal.wt     : num [1:9] 4.84 4.7 4.9 3.92 4.6 ...
   $ Territoriality      : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 1 1 2 2
```

```
Dropped taxa:
   perissodactyla.tree { 9 ( 9 } 0 ) perissodactyla.data
```

```
> # but this can be turned off
> (perissoFull <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial, na.omit=FAl
```

```
Comparative dataset of 13 taxa:
Phylogeny: perissodactyla.tree
   13 tips, 11 internal nodes
   chr [1:13] "Dicerorhinus sumatrensis" "Rhinoceros sondaicus" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:13] 5.91 6.15 6.24 6.18 6.26 ...
   $ log.gestation.length: num [1:13] 2.6 2.68 2.68 2.65 2.69 ...
   $ log.neonatal.wt     : num [1:13] 4.54 4.7 4.84 4.7 4.9 ...
   $ Territoriality      : Factor w/ 2 levels "No","Yes": NA NA 2 1 2 NA NA 2 2 1 ...
Dropped taxa:
   perissodactyla.tree { 5 ( 13 } 0 ) perissodactyla.data
```

```
> na.omit(perisso)
```

```
Comparative dataset of 9 taxa:
Phylogeny: perissodactyla.tree
   9 tips, 7 internal nodes
   chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:9] 6.24 6.18 6.26 5.46 5.65 ...
   $ log.gestation.length: num [1:9] 2.68 2.65 2.69 2.6 2.59 ...
   $ log.neonatal.wt     : num [1:9] 4.84 4.7 4.9 3.92 4.6 ...
   $ Territoriality      : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 1 1 2 2
Dropped taxa:
   perissodactyla.tree { 9 ( 9 } 0 ) perissodactyla.data
```

The `scope` argument uses a model formula to indicate which variables to consider when reducing a comparative data object to complete rows and can be used directly by `na.omit` or indirectly through `comparative.data`.

```
> comparative.data(perissodactyla.tree, perissodactyla.data, Binomial, scope= log.female.wt ~ log.
```

```
Comparative dataset of 12 taxa:
Phylogeny: perissodactyla.tree
   12 tips, 10 internal nodes
   chr [1:12] "Dicerorhinus sumatrensis" "Rhinoceros sondaicus" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:12] 5.91 6.15 6.24 6.18 6.26 ...
   $ log.gestation.length: num [1:12] 2.6 2.68 2.68 2.65 2.69 ...
   $ log.neonatal.wt     : num [1:12] 4.54 4.7 4.84 4.7 4.9 ...
   $ Territoriality      : Factor w/ 2 levels "No","Yes": NA NA 2 1 2 NA 2 2 1 1 ...
Dropped taxa:
   perissodactyla.tree { 6 ( 12 } 0 ) perissodactyla.data
Scope of complete data:
 log.female.wt ~ log.gestation.length
```

```
> na.omit(perissoFull, scope=log.female.wt ~ log.gestation.length + Territoriality)
```

```
Comparative dataset of 9 taxa:
Phylogeny: perissodactyla.tree
   9 tips, 7 internal nodes
   chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:9] 6.24 6.18 6.26 5.46 5.65 ...
   $ log.gestation.length: num [1:9] 2.68 2.65 2.69 2.6 2.59 ...
   $ log.neonatal.wt     : num [1:9] 4.84 4.7 4.9 3.92 4.6 ...
   $ Territoriality      : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 1 1 2 2
Dropped taxa:
   perissodactyla.tree { 9 ( 9 } 0 ) perissodactyla.data
Scope of complete data:
 log.female.wt ~ log.gestation.length + Territoriality
```

2.1.2 subset

This method functions like the `subset` function for data frames: the `subset` argument is used to specify a logical subset of rows using one or more of the data columns; the `select` argument can be used to choose which variables to include in the subset.

```
> subset(perissoFull, subset=! is.na(Territoriality))
```

```
Comparative dataset of 9 taxa:
Phylogeny: perissodactyla.tree
   9 tips, 7 internal nodes
   chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:9] 6.24 6.18 6.26 5.46 5.65 ...
   $ log.gestation.length: num [1:9] 2.68 2.65 2.69 2.6 2.59 ...
   $ log.neonatal.wt     : num [1:9] 4.84 4.7 4.9 3.92 4.6 ...
   $ Territoriality      : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 1 1 2 2
Dropped taxa:
   perissodactyla.tree { 9 ( 9 } 0 ) perissodactyla.data
```

```
> subset(perissoFull, subset=log.female.wt > 5.5, select=c(log.female.wt, log.gestation.length))
```

```
Comparative dataset of 6 taxa:
Phylogeny: perissodactyla.tree
   6 tips, 5 internal nodes
   chr [1:6] "Dicerorhinus sumatrensis" "Rhinoceros sondaicus" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:6] 5.91 6.15 6.24 6.18 6.26 ...
   $ log.gestation.length: num [1:6] 2.6 2.68 2.68 2.65 2.69 ...
Dropped taxa:
   perissodactyla.tree { 12 ( 6 } 0 ) perissodactyla.data
```

2.1.3 [

This method treats the dataset as the rows (taxa) and columns (variables) of a matrix. The first index allows taxa to be specified by name or by their position in the sequence of tip labels for the phylogeny. The second index selects variables in the same way as in the extract method ([.data.frame) for a data frame.

```
> horses <- grep('Equus', perissoFull$phy$tip.label)
> perissoFull[horses,]


Comparative dataset of 5 taxa:
Phylogeny: perissodactyla.tree
   5 tips, 3 internal nodes
   chr [1:5] "Equus grevyi" "Equus burchelli" "Equus zebra" "Equus africanus" ...
Data: perissodactyla.data
   $ log.female.wt       : num [1:5] 5.65 5.48 5.46 5.44 5.46
   $ log.gestation.length: num [1:5] 2.59 2.56 2.56 2.56 2.56
   $ log.neonatal.wt     : num [1:5] 4.6 4.48 4.4 4.4 4.4
   $ Territoriality      : Factor w/ 2 levels "No","Yes": 2 1 1 2 2
Dropped taxa:
   perissodactyla.tree { 13 ( 5 } 0 ) perissodactyla.data


> perissoFull[horses, 2:3]


Comparative dataset of 5 taxa:
Phylogeny: perissodactyla.tree
   5 tips, 3 internal nodes
   chr [1:5] "Equus grevyi" "Equus burchelli" "Equus zebra" "Equus africanus" ...
Data: perissodactyla.data
   $ log.gestation.length: num [1:5] 2.59 2.56 2.56 2.56 2.56
   $ log.neonatal.wt     : num [1:5] 4.6 4.48 4.4 4.4 4.4
Dropped taxa:
   perissodactyla.tree { 13 ( 5 } 0 ) perissodactyla.data
```

Far more functionality for creating and manipulating comparative datasets is provided by the `phylobase` package. It is intended to incorporate this functionality into `caper`.

2.2 Example datasets

The examples in this vignette make use of the following datasets:

`shorebird` This dataset includes a phylogeny of 71 species of shorebirds along with data on male and female body mass, egg mass, clutch size and mating system (Lislevand and Thomas, 2006).

`syrphidae` This dataset includes a phylogeny of 204 genera of hoverfly along with data on the species richness of each of those genera (Katzourakis et al., 2001).

`perissodactyla` This dataset includes a phylogeny of 18 perissodactyl species and a data frame of female and neonatal mass, gestation length and territoriality for 13 of those species. The data frame contains missing data and the dataset was provided as an example with the original CAIC program (Purvis and Rambaut, 1995b).

`IsaacEtAl` The datafile contains species level phylogenies and data sets of nine variables for each of four mammalian orders (Primates, Carnivora, Chiroptera and Marsupialia). The data were published in supplementary material for Isaac et al. (2005). The variables are body mass, age at sexual maturity, gestation length, interbirth interval, litter size , population density, group size, mass dimorphism and length.dimorphism. The data sets all contain missing data and all the variables have been natural log transformed.

`British Birds` The datafile contains a species level phylogeny for 249 species of British birds (some represented by congeneric surrogates) and a data set on the conservation status of 181 of those species (Thomas, 2008).

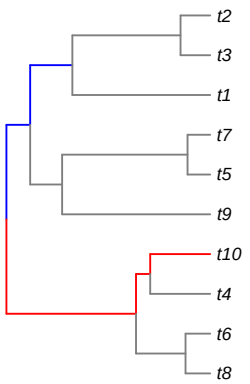
3 Methods and functions provided by `caper`.

The `caper` package provides two broad sets of methods of accounting for phylogenetic autocorrelation in analyses. One is the use of phylogenetic independent contrasts (Felsenstein, 1985), the other is the use of a phylogenetic variance-covariance matrix within a standard linear model (Freckleton et al., 2002) [other refs]. These two approaches are basically equivalent, although small differences may arise from the way in which polytomies are handled.

3.0.1 Phylogenetic linear models

In a basic linear regression, the response variable (y) is modelled as the product of the estimated coefficients (β) and the explanatory variables ($\mathbf{X}$), plus residual variation (ϵ): $y = \beta\mathbf{X} + \epsilon$. If the cases in the model are related taxa, then the values in y and $\mathbf{X}$ are no longer independent: each value will be more similar to some values (closer relatives) than others (distant relatives).

The `pgls` function addresses this problem by incorporating the covariance between taxa into the calculation of estimated coefficients: this is a generalized least squares (GLS) model. The covariance matrix ($\mathbf{V}$), showing the expected covariance between each pair of tips is calculated using the branch lengths of a phylogeny showing how the taxa are related (Fig. 2).



$$\begin{bmatrix}
\mathbf{9.0} & 7.7 & 2.9 & 1.0 & 1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
7.7 & \mathbf{9.0} & 2.9 & 1.0 & 1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
2.9 & 2.9 & \mathbf{9.0} & 1.0 & 1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
1.0 & 1.0 & 1.0 & \mathbf{9.0} & 8.0 & 2.5 & 0.0 & 0.0 & 0.0 & 0.0 \\
1.0 & 1.0 & 1.0 & 8.0 & \mathbf{9.0} & 2.5 & 0.0 & 0.0 & 0.0 & 0.0 \\
1.0 & 1.0 & 1.0 & 2.5 & 2.5 & \mathbf{9.0} & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \mathbf{9.0} & 6.3 & 5.7 & 5.7 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 6.3 & \mathbf{9.0} & 5.7 & 5.7 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 5.7 & 5.7 & \mathbf{9.0} & 7.9 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 5.7 & 5.7 & 7.9 & \mathbf{9.0}
\end{bmatrix}$$

Figure 2: A phylogenetic tree of 10 taxa and the variance covariance matrix ($\mathbf{V}$) of that phylogeny. The diagonal of the matrix (bold values) shows the path length from each tip to the root (example in red). Off diagonal values show the shared path length for a given pair of tips (example in blue).

The initial covariance matrix ($\mathbf{V}$) taken from an ultrametric phylogeny assumes a Brownian model of evolution: that variation between tips accumulates along all branches of the tree at a rate proportional to the length of the branches. However, not all variables necessarily correspond to this assumption and likelihood methods can be used to find transformations of $\mathbf{V}$ that improve the fit of the model to the data. Three transformations are implemented in the `pgls` function (Fig. 3):

lambda (λ) The internal branch lengths of the phylogeny are multiplied by a constant. When $\lambda = 0$, internal branch lengths are all zero and all the variation in the data is modelled as a function of the independent evolution along the branches leading to the tips. If $\lambda > 1$, then the data shows more covariance than expected under a Brownian model.

Note that a `pgls` model with a lambda of 0 will not be the same as a standard linear model if the tree is not ultrametric: although the covariance terms will all be zero, the variances of the tips will differ.

delta (δ) In this case, all the values in the $\mathbf{V}$ matrix are raised to the power of δ. This is a transformation of the sum of the length of the shared branches between two tips l: $\sum (l)^{\delta}$.

kappa (κ) All branch lengths in the phylogeny are raised to the power κ — the elements of the **V** are the sum of the individually transformed branch lengths $\sum(l^{\kappa})$. As κ decrease towards zero, all branches will become of equal length, representing an evolutionary model where variation accumulates only when two clades diverge.

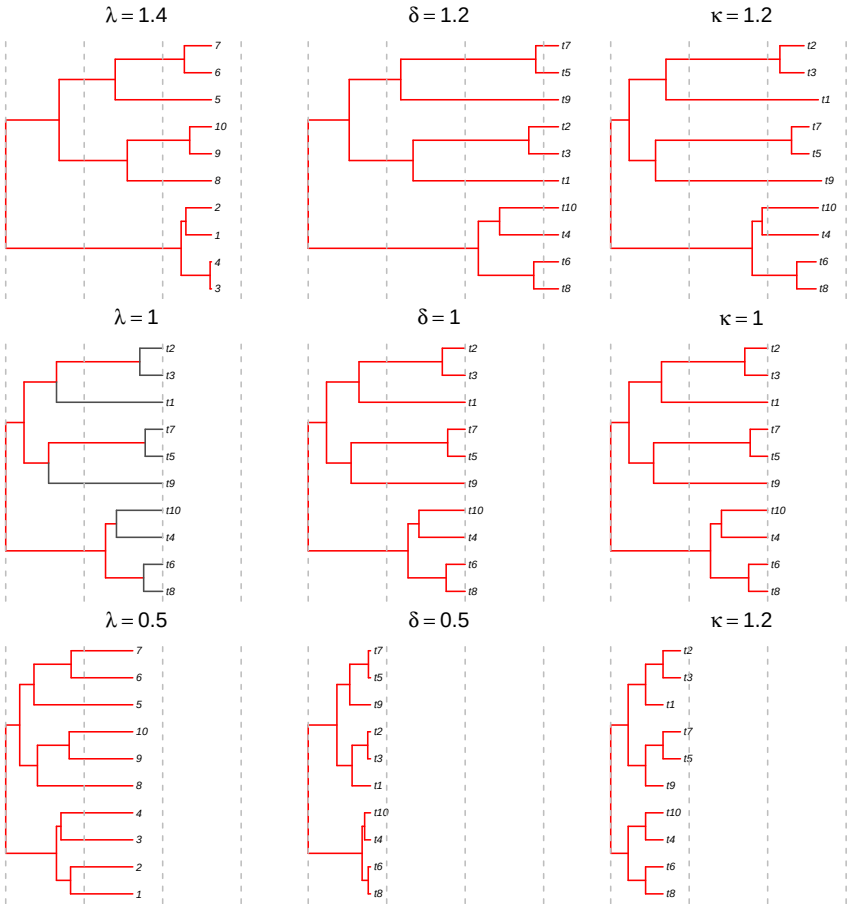


Figure 3: Examples of λ, δ and κ branch length transformations. The branches affected by a given transformation are shown in red.

3.0.2 Fitting phylogenetic GLS models: `pgls`

The `pgls` function implements GLS models accounting for phylogeny. It also includes the three branch length transformations (λ, κ, δ) described above. The values of these may be fixed at values provided by the user or optimised within bounds to find the maximum likelihood transformation. Multiple transformations can be optimised simultaneously, although the underlying evolutionary model may be hard to interpret.

The function uses a formula interface to describe the model and uses data provided as a comparative data object. Because the model fitting uses a covariance matrix, which can be computationally expensive to extract from a phylogeny, it is useful to include this when creating the comparative data object at the start of an analysis using `vcv=TRUE`. The resulting `pgls` model object has standard `print` and `summary` methods.

```
> data(shorebird)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE)
> mod <- pgls(log(Egg.Mass) ~ log(M.Mass), shorebird)
> print(mod)
```

```
Call:
pgls(formula = log(Egg.Mass) ~ log(M.Mass), data = shorebird)

Coefficients:
(Intercept)  log(M.Mass)
    -0.4218       0.6756


> summary(mod)


Call:
pgls(formula = log(Egg.Mass) ~ log(M.Mass), data = shorebird)

Residuals:
      Min        1Q    Median        3Q       Max
-0.104152 -0.018160  0.006679  0.022493  0.078391

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [Fix]  : 1.000
delta  [Fix]  : 1.000

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.421775   0.227825 -1.8513   0.0684 .
log(M.Mass)  0.675619   0.034791 19.4194   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03343 on 69 degrees of freedom
Multiple R-squared: 0.8453,        Adjusted R-squared: 0.8431
F-statistic: 377.1 on 1 and 69 DF,  p-value: < 2.2e-16
```

3.1 Optimising branch length transformations: `profile.pgls`.

The `pgls` function has three arguments to specify parameter values for the λ, κ and δ transformations. However, the value for these parameters can alternatively be specified as `"ML"`, in which case the maximum likelihood value for each ML parameter will be found within set bounds. The default bounds ($\lambda = [1 \times 10^{-6}, 1], \kappa = [1 \times 10^{-6}, 3], \delta = [1 \times 10^{-6}, 3]$) may be altered but cannot be negative. Optimising multiple parameters can lead to computational problems with lower bounds very close to zero, so it may be necessary to increase the lower bound slightly.

Because calculating likelihood under a κ transformation requires a transformation of the individual branch lengths, it is not possible with a simple $\mathbf{V}$ matrix. This is because $\mathbf{V}$ only holds the sums of those branch lengths. Any variation of κ, whether for optimization or for profiling likelihood values, therefore needs a 3 dimensional $\mathbf{V}$ array, in which the branch lengths are held separately. The `vcv.dim` argument to the `comparative.data` function creates this.

By default, `pgls` will use likelihood ratio tests to establish confidence bounds on optimised parameters. The likelihood value at the ML estimate is tested against the values at the parameter bounds and confidence intervals of the ML estimate are also calculated. The `profile.pgls` function allows the likelihood surface of a parameter for a model to be displayed. If the ML estimate of the parameter is known then then the profile will show this along with confidence limits (Fig. 4). The likelihood profile is taken for a single parameter at a time, with the other two parameters held at their fixed or optimum values.

```
> data(shorebird)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE, vcv.dim=3)
> mod <- pgls(log(Egg.Mass) ~ log(M.Mass), shorebird, lambda='ML')
> summary(mod)


Call:
pgls(formula = log(Egg.Mass) ~ log(M.Mass), data = shorebird,
    lambda = "ML")

Residuals:
      Min        1Q    Median        3Q       Max
-0.079306 -0.015127  0.005864  0.019562  0.065373

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.956
   lower bound : 0.000, p = < 2.22e-16
   upper bound : 1.000, p = 0.047331
   95.0% CI   : (0.867, 1.000)
delta  [Fix]  : 1.000

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.486129   0.205686 -2.3635  0.02093 *
log(M.Mass)  0.688645   0.032647 21.0937  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.02905 on 69 degrees of freedom
Multiple R-squared: 0.8657,        Adjusted R-squared: 0.8638
F-statistic: 444.9 on 1 and 69 DF,  p-value: < 2.2e-16


> mod.l <- pgls.profile(mod, 'lambda')
> mod.d <- pgls.profile(mod, 'delta')
> mod.k <- pgls.profile(mod, 'kappa')
> plot(mod.l); plot(mod.d); plot(mod.k)
```

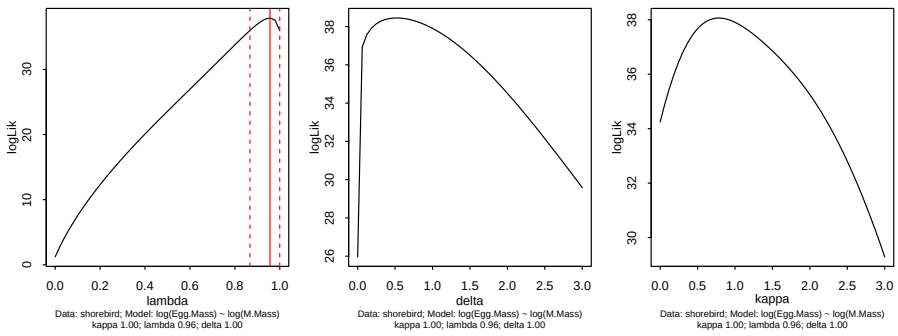


Figure 4: Likelihood surfaces of λ, κ and δ for a simple model (log Egg.Mass log M.Mass) on the `shorebird` dataset. The ML estimate has only been fitted for λ and so this is the only plot that shows confidence limits.

3.1.1 Criticism and simplification of `pgls` models: `plot`, `anova` and `AIC`.

The `caper` package provides standard generic methods for examining `pgls` models. The first is the `plot` methods, which produces four diagnostic plots. The first two show the fit of the residuals to a normal distribution: a density plot of the distribution of the residuals and a normal Q-Q plot the distribution of the residuals against their expected distribution under a normal distribution. The second two show the fitted values against both the residuals and the observed value to look for pattern in the residuals within the model. Figure 5 shows these plots for the model above using the command `plot(mod)`.

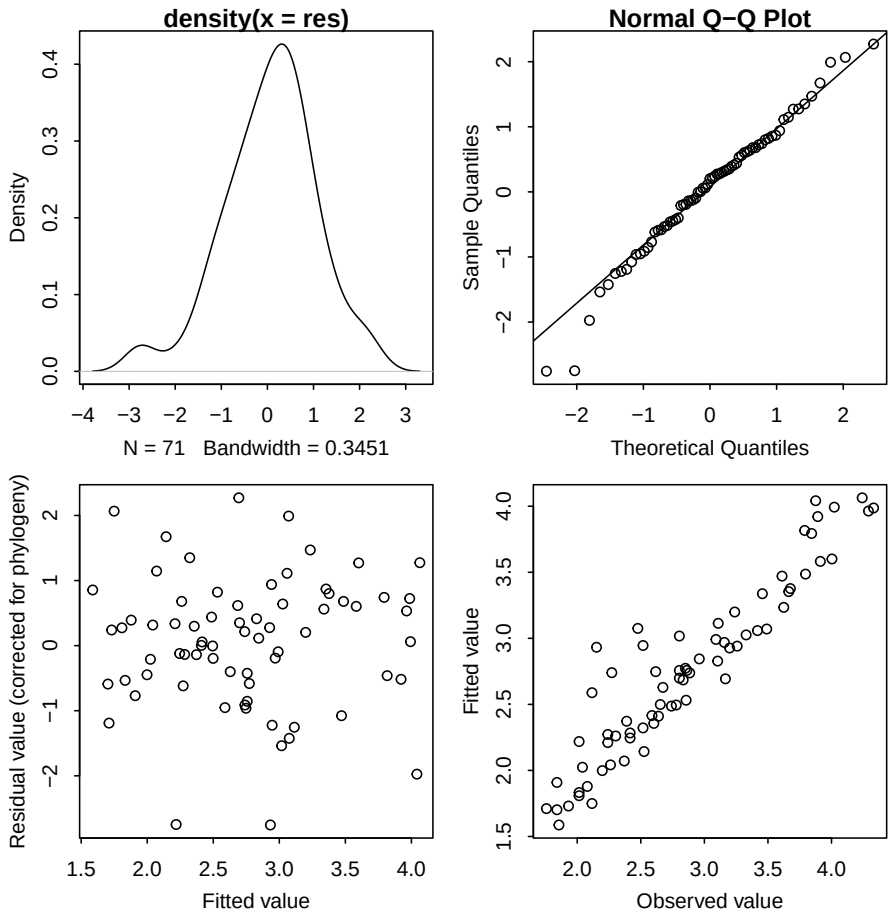


Figure 5: Model diagnostic plots for a simple model (`log(Egg.Mass) ~log(M.Mass)`) on the `shorebird` dataset.

The package also provides `anova` and `logLik` methods that allow model comparison. The `logLik` method allows the generic `AIC` methods to be used with `pgls` models. Analysis of variance tables for a `pgls` model uses sequential sums of squares: each F value is the marginal improvement in the model given the combined effects of the terms above. The `anova` command can also be used to compare a set of models.

```
> mod1 <- pgls(log(Egg.Mass) ~ log(M.Mass) * log(F.Mass), shorebird)
> anova(mod1)


Analysis of Variance Table
Sequential SS for pgls: lambda = 1.00, delta = 1.00, kappa = 1.00

Response: log(Egg.Mass)
                       Df  Sum Sq Mean Sq  F value Pr(>F)
```

```
log(M.Mass)              1 0.42146 0.42146 371.8424 <2e-16 ***
log(F.Mass)              1 0.00114 0.00114   1.0014 0.3206
log(M.Mass):log(F.Mass)  1 0.00004 0.00004   0.0339 0.8545
Residuals               67 0.07594 0.00113
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> mod2 <- pgls(log(Egg.Mass) ~ log(M.Mass) + log(F.Mass), shorebird)
> mod3 <- pgls(log(Egg.Mass) ~ log(M.Mass) , shorebird)
> mod4 <- pgls(log(Egg.Mass) ~ 1, shorebird)
> anova(mod1, mod2, mod3, mod4)
```

```
Analysis of Variance Table
pgls: lambda = 1.00, delta = 1.00, kappa = 1.00

Model 1: log(Egg.Mass) ~ log(M.Mass) * log(F.Mass)
Model 2: log(Egg.Mass) ~ log(M.Mass) + log(F.Mass)
Model 3: log(Egg.Mass) ~ log(M.Mass)
Model 4: log(Egg.Mass) ~ 1
  Res.Df     RSS Df Sum of Sq        F Pr(>F)
1     67 0.07594
2     68 0.07598 -1  -0.00004   0.0339 0.8545
3     69 0.07711 -1  -0.00114   1.0014 0.3206
4     70 0.49858 -1  -0.42146 371.8424 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> AIC(mod1, mod2, mod3, mod4)
```

```
     df       AIC
mod1  4 -64.96854
mod2  3 -66.93263
mod3  2 -67.87981
mod4  1  62.63955
```

3.2 Phylogenetic independent contrasts

The `caper` package implements the methods originally provided in the programs CAIC (Purvis and Rambaut, 1995b) and MacroCAIC (Agapow and Isaac, 2002). Both programs calculate phylogenetically independent contrasts in a set of variables and then use linear models of those contrasts to test for evolutionary relationships. All of the functions in this section require a comparative dataset object and a description of a linear model as an R formula (see `formula()`).

In contrast model functions, a reference variable (`ref.var`) may also be provided. This is used to standardize the directions in which contrasts are calculated in multivariate models. If no reference variable is provided, the function defaults to using the first explanatory variable specified in the model formula. All contrast model functions enforce regression through the origin (Garland et al., 1992).

3.2.1 Variable names in contrast functions

The three functions below (`crunch`, `brunch`, `macrocaic`) all make use of existing methods within R to generate a model matrix. In normal linear models, the terms in the model formula allow

transformations to be specified within models: a model can use `y ~ log(x)` and subsequent modifications of the model will expect to be able to use a data frame containing `x` that will be logged before use. These contrast functions do support this in initial model fitting. However modifications cannot be made so simply to contrast models: the variable `log(x)` now represents *contrasts* in `log(x)`, and reference back to the original variable `x` will not yield the correct results. Generating consistent behaviour using the R modelling framework in these cases is not easy.

It is therefore strongly recommended that any transformation of data should be carried out in the original data frame and that these transformed variables are used in model specification.

3.2.2 Continuous variables: `crunch`

The `crunch` function provides calculation of independent contrasts (Felsenstein, 1985) for continuous variables, as implemented in the 'crunch' option of the CAIC package (Purvis and Rambaut, 1995b), using the method of Pagel (1992) to calculate contrasts at polytomies. The following example shows an example analysis using the `shorebird` dataset.

```
> data(shorebird)
> shorebird.data$lgEgg.Mass <- log(shorebird.data$Egg.Mass)
> shorebird.data$lgM.Mass <- log(shorebird.data$M.Mass)
> shorebird.data$lgF.Mass <- log(shorebird.data$F.Mass)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
> crunchMod <- crunch(lgEgg.Mass ~ lgF.Mass + lgM.Mass, data=shorebird)
> summary(crunchMod)


Call:
lm(lgEgg.Mass ~ lgF.Mass + lgM.Mass - 1, data = contrData)

Residuals:
      Min        1Q    Median        3Q       Max
-0.075470 -0.019982  0.005309  0.016856  0.083748

Coefficients:
         Estimate Std. Error t value Pr(>|t|)
lgF.Mass  -0.2203     0.2349  -0.938 0.352039
lgM.Mass   0.8926     0.2370   3.767 0.000379 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03503 on 60 degrees of freedom
Multiple R-squared:  0.8476,        Adjusted R-squared:  0.8425
F-statistic: 166.8 on 2 and 60 DF,  p-value: < 2.2e-16
```

3.2.3 Categorical variables: `brunch`

The inclusion of categorical variables in phylogenetic independent contrast models is problematic. Continuous variables are assumed to evolve under a Brownian process, which permits estimates of nodal values and hence allows nested contrasts to be drawn. There is no sensible model for reconstructing the nodal values of a categorical variable, except where all daughters share the same categorical value. However, identifying nodes with daughter variables that differ in the value of the categorical variable permit independent contrasts to be identified (Burt, 1989). While nodal estimates of continuous variables can be made below at nodes below a `brunch` contrast, to do so would assume independent evolution of these variables at the nodes of any deeper contrasts (Burt, 1989). As a result, the `brunch` algorithm only uses the data from any single tip in the calculation of a single contrast.

The example below shows `brunch` contrasts for the `perissodactyla` dataset calculated using territoriality. Figure 6 shows how these contrasts are extracted from the available data. In each case, a contrast is drawn between groups of species showing differing territorial behaviour and no species is used in more than one contrast.

```
> perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial)
> brunchMod <- brunch(log.female.wt ~ Territoriality, data=perisso)
> caic.table(brunchMod)
```

```
   log.female.wt Territoriality contrVar nChild nodeDepth nodeAge studentResid
19   0.055861436              1  2.00000      2         2       1  -0.04153674
22  -0.003521142              1 10.66667      3         3       8  -1.57705483
24   0.124450793              1  2.00000      2         2       1   1.91107764
```

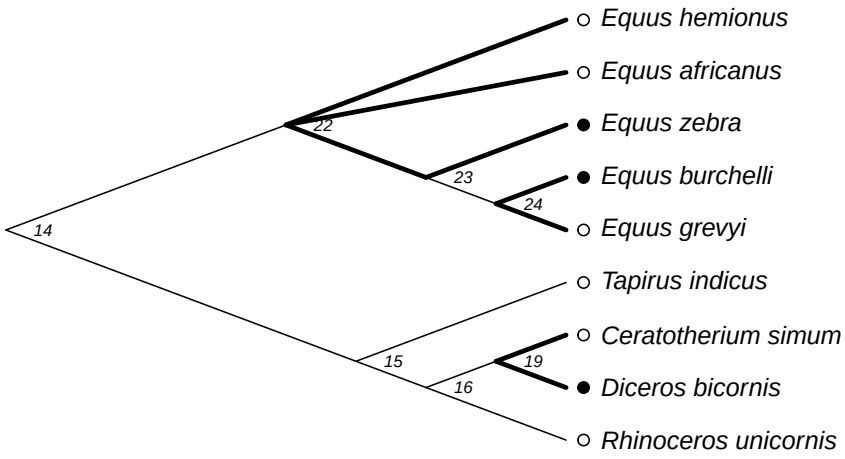


Figure 6: Location of `brunch` contrasts for territoriality in the `perissodactyla` dataset. Branches in bold show the tips contributing to each contrast. See also the output of `caic.table` in the main text.

3.2.4 Species richness contrasts: `macrocaic`

One common approach in macroevolution is to investigate the links between the traits of taxa and their species richness. It is inappropriate to calculate `crunch` contrasts in species richness, particularly for nested nodes, where `crunch` contrasts would estimate nodal values as a weighted average of species richness for the daughter taxa, rather than tracking total species richness. For this reason, Agapow and Isaac (2002) developed the program MacroCAIC, which implemented the calculation of species richness contrasts.

The MacroCAIC program, and the `macrocaic` function, take a statistical model of species richness as a function of continuous explanatory variables and calculate `crunch` contrasts in the explanatory variables and one of two recommended species richness contrasts as a response (Isaac et al., 2003). These two contrasts are the relative rate difference (RRD: $ln(N_1/N_2)$) and the proportion dominance index (PDI: $(N_1/(N_1 + N_2)) - 0.5$), where N_1 and N_2 are the number of species in the daughter clades and N_1 is the richness of the clade with the higher value in the focal explanatory variable (Agapow and Isaac, 2002).

```
> data(IsaacEtAl)
> primates <- comparative.data(primates.tree, primates.data, binomial, na.omit=FALSE)
> primatesBodySize <- macrocaic(species.rich ~ body.mass, data=primates)
> summary(primatesBodySize)
```

```
Call:
lm(species.rich ~ body.mass - 1, data = contrData)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2046 -1.0206 -0.3174  0.7218  3.1996

Coefficients:
          Estimate Std. Error t value Pr(>|t|)
body.mass  -0.2348     0.6275  -0.374    0.709

Residual standard error: 1.305 on 91 degrees of freedom
Multiple R-squared:  0.001536,        Adjusted R-squared:  -0.009436
F-statistic:  0.14 on 1 and 91 DF,  p-value: 0.7092


>
```

3.2.5 Phylogenetic signal: `phylo.d`

Fritz and Purvis (2010) proposed the statistic D as a measure of phylogenetic signal in binary traits. The D statistic makes use of the way in which estimated nodal values calculated lusing `crunch` change along the edges of a phylogeny. If a trait is highly conserved, with only a basal division between two clades expressing either trait value, then the only change will be along the edges between the two daughter clades at the root. This will give a summed value of 1: the two differences between the root nodal value of 0.5 and the ancestors of the 1 and 0 clades. In contrast, if the trait is labile, more differences will be observed and the sum will be higher.

The sum of these changes across a phylogeny will be governed by the phylogenetic signal in the variable, but also by the size of the phylogeny and by the relative proportions of the two states (prevalence). In order to standardise the effects of phylogeny size and prevalence, `phylo.d` uses two simulated null models:

Phylogenetic randomness Trait values are randomly shuffled relative to the tips of the phylogeny.

Brownian threshold model A continuous trait is evolved along the phylogeny under a Brownian process and then converted to a binary trait using a threshold that reproduces the relative prevalence of the observed trait.

The mean sum of changes under the Brownian ($\bar{s_b}$) and random ($\bar{s_r}$) models are used to calibrate the observed sum of changes (s_o) values: $D = (s_o - \bar{s_b})/(\bar{s_r} - \bar{s_b})$ (Fig 7). On this standardised scale, a variable with random association will have $D \approx 1$ since $s_o \approx \bar{s_r}$ and a variable following the Brownian model will have $D \approx 0$ since $s_o = \bar{s_b}$. Values of D smaller than 0 are phylogenetically more conserved than under the Brownian model and values of D greater than 1 are phylogenetically overdispersed. The simulated values can also be used to assess whether the observed D differs significantly from the simulated models.

The example below uses the `BritishBirds` dataset to calculate D for the phylogenetic distribution of 'Red' conservation status for 181 species of British birds (Fritz and Purvis, 2010; Thomas, 2008).

```
> data(BritishBirds)
> BritishBirds <- comparative.data(BritishBirds.tree, BritishBirds.data, binomial)
> redPhyloD <- phylo.d(BritishBirds, binvar=Red_list)
> print(redPhyloD)
```

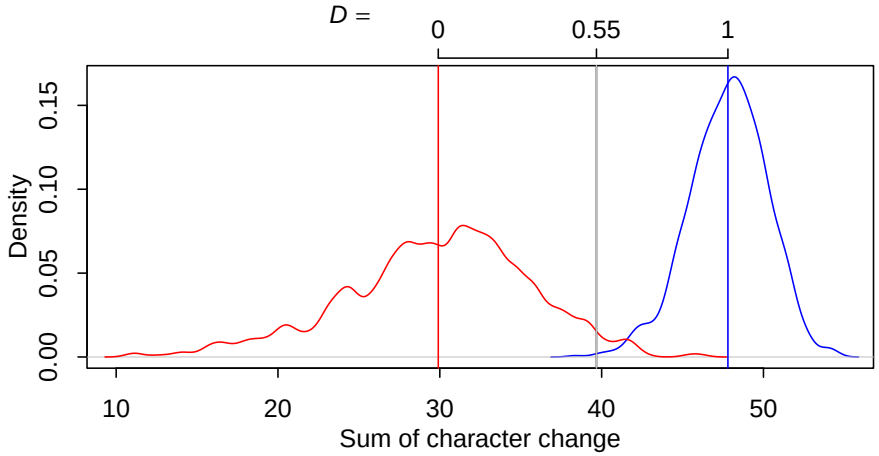


Figure 7: Scaling of D values for the observed sum of character change from the distributions of simulated sums of character change under models of random association (blue line) and a Brownian process (red line). The distributions of the simulations are used to test the significance of departures from either model.

```
Calculation of D statistic for the phylogenetic structure of a binary variable

  Data :  BritishBirds.data
  Binary variable :  Red_list
  Counts of states:  0 = 149
                     1 = 32
  Phylogeny :  BritishBirds.tree
  Number of permutations :  1000

Estimated D :  0.5599617
Probability of E(D) resulting from no (random) phylogenetic structure :  0.003
Probability of E(D) resulting from Brownian phylogenetic structure    :  0.022
```

3.3 Checking and comparing contrast models.

The three contrast model functions (`crunch`, `brunch` and `macrocaic`) all return an object of class `caic` for which a number of common functions and methods are provided, including `print` and `summary` methods. These are useful for assessing the robustness of contrast models. The simplest function is `caic.table`, which returns a table of contrasts and nodal values that is very like the original contrast files provided by CAIC and MacroCAIC. This data frame (see the example in the `brunch` section, page 13) provides all the information needed to assess model suitability but some convenience functions are provided for standard checks.

3.3.1 Testing evolutionary assumptions: `caic.diagnostics`.

This function implements three diagnostic regression tests for the robustness of a contrast model (Garland et al., 1992; Purvis and Rambaut, 1995a). The function performs regression for each test and optionally displays scatterplots of the diagnostic test data. The tests examine the behaviour of the absolute standardised contrasts with the scale of the following nodal values:

Estimated nodal value This tests an evolutionary assumption of the contrast model: that there is no relationship between the magnitude of estimated contrasts and the estimated nodal value. Put another way, the proportional difference between daughter clades should be independent of magnitude of their trait values.

Estimated standard deviation at the node The evolutionary model underlying contrasts (Felsenstein, 1985) assumes that variance accumulates in continuous characters equally along all branches and that the amount of variation is proportional to branch length. The square root of the expected variance calculated at each node can therefore be used to standardize the absolute difference ('raw contrasts') between sister taxa. If this standardization is successful, there should be no relationship between absolute values of standardized contrasts and the estimated standard deviation.

Age of the node If the phylogeny is ultrametric, then the node age, as time from the present, provides a further test of the effectiveness of the standardization of the contrasts.

If any of these diagnostic regressions are significant, then the assumption of evolution under a Brownian walk has been violated and a transformation of the data or branch lengths may be required. As in the example use below on the `shorebird` dataset, log transformation of continuous variables is often an effective solution for problems with contrast diagnostics (Fig 8).

```
> par(mfrow=c(2,3))
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
> crunchMod <- crunch(Egg.Mass ~  M.Mass, data=shorebird)
> caic.diagnostics(crunchMod)


M.Mass :
    Estimate Std. Error t value  Pr(>|t|)
NV  0.065449   0.010770  6.0769 9.138e-08 ***
SD  0.700158   1.468213  0.4769    0.6352
AGE 3.258202   2.809928  1.1595    0.2508

---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


> shorebird.data$lgEgg.Mass <- log(shorebird.data$Egg.Mass)
> shorebird.data$lgM.Mass <- log(shorebird.data$M.Mass)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
> crunchMod2 <- crunch(lgEgg.Mass ~  lgM.Mass, data=shorebird)
> caic.diagnostics(crunchMod2)


lgM.Mass :
      Estimate Std. Error t value Pr(>|t|)
NV  -0.0103218  0.0097468 -1.0590   0.2938
SD  -0.0033931  0.0054315 -0.6247   0.5345
AGE  0.0090846  0.0104824  0.8666   0.3896

---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3.2 Robust contrasts: `caic.robust`

In Figure 8, there is a systematic problem with the unlogged model and, although one contrast has a high studentized residual, this point does not drive the relationship. In other cases, individual contrasts may exert undue influence over an otherwise well behaved set of contrasts. This is more commonly the case for models where there are known to be problems with branch length estimates or where all branch lengths are treated as equal length. In such cases, these nodes should be removed from the set of contrasts used in the linear model. A threshold of absolute studentized residuals greater than 3 is commonly applied (Jones and Purvis, 1997; Garland et al., 1992).

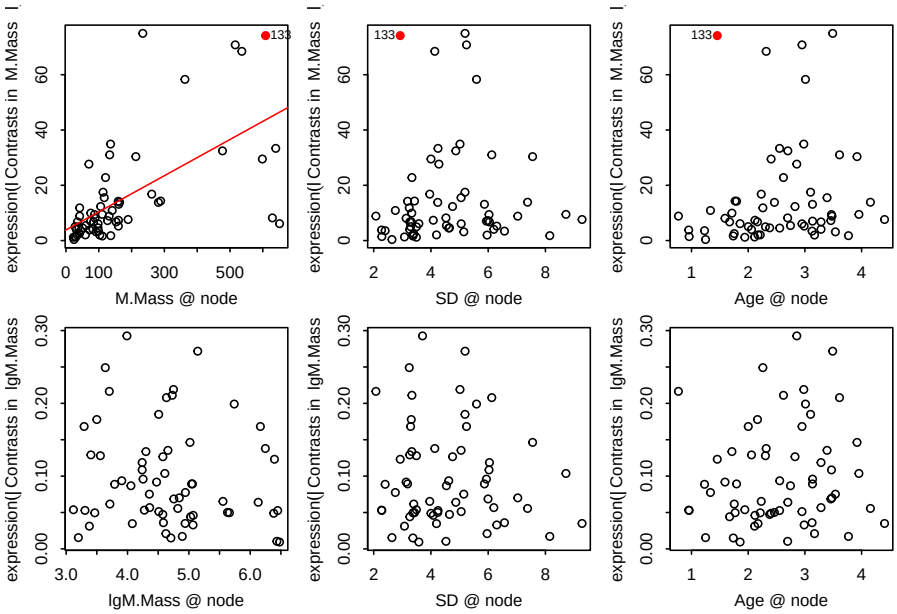


Figure 8: Diagnostic plots for contrasts in male shorebird mass for contrast models on unlogged (top row, poorly distributed with one serious outlier) and logged (bottom row) data.

The contrast model functions in `caper` provide the `robust` argument to set the size of this threshold. By default, this is set to infinity: all contrasts are used to fit the linear model. This can be changed by altering the `robust` argument but can also be modified after initial inspection of contrast diagnostics using the `caic.robust` function. This simple method alter the threshold for a `caic` object and returns a modified version with this threshold filter enforced.

Note that none of the contrasts or residuals are recalculated using the `robust` options: the `caic` object retains all of the original contrast data and the studentized residuals are *always* those obtained from the full set of contrasts. The threshold filter is only applied in two contexts:

1. The statistical model of the contrasts held in the `caic` object and displayed by the model summary.
2. The points displayed by `caic.diagnostics`. Note that the `outlier` argument to this function is independent of the value of `robust` for an object.

Although clearly a bad model, the example below uses unlogged data from `shorebird` to illustrate the use of the functions (Fig. 9).

```
> data(shorebird)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
> crunchMod <- crunch(Egg.Mass ~ M.Mass, data=shorebird)
> caic.diagnostics(crunchMod)


M.Mass :
    Estimate Std. Error t value  Pr(>|t|)
NV  0.065449   0.010770  6.0769 9.138e-08 ***
SD  0.700158   1.468213  0.4769    0.6352
AGE 3.258202   2.809928  1.1595    0.2508

---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> crunchModRobust <- caic.robust(crunchMod)
> caic.diagnostics(crunchModRobust, outlier=2)

Excluding 1 contrast with absolute studentised residuals > 3

M.Mass :
    Estimate Std. Error t value  Pr(>|t|)
NV  0.057357   0.010998  5.2154 2.474e-06 ***
SD  1.348589   1.353080  0.9967   0.32299
AGE 4.920180   2.571071  1.9137   0.06052 .

---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

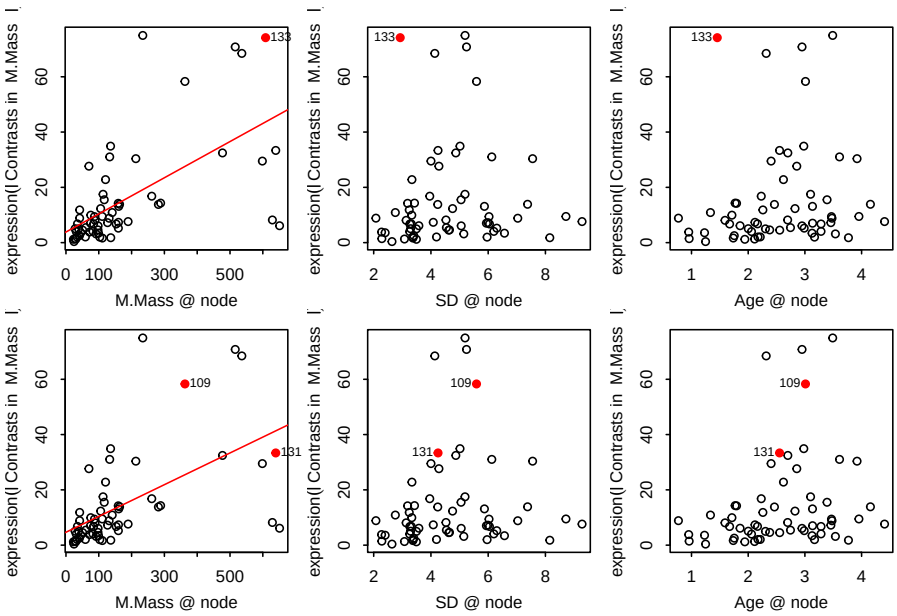


Figure 9: Diagnostic plots and `caic.robust`: the upper row shows diagnostic plots for the full model, highlighting a single point with an absolute studentized residual > 3. The lower row shows the same plots after applying `caic.robust`: the highlighted point has been excluded and the argument `outlier` has been used to reveal points with absolute residuals > 2.

3.3.3 Model criticism: `plot`

The preceding tools are broadly useful for assessing whether the evolutionary assumptions of contrast modelling have been met and for excluding problematic nodes. However it is still wise to check whether the statistical assumptions of the linear model on the contrasts are met. The `caper` package provides a simple wrapper to allow the standard model criticism plots to be generated for a `caic` object (Fig 10).

```
> data(shorebird)
> shorebird.data$lgEgg.Mass <- log(shorebird.data$Egg.Mass)
> shorebird.data$lgM.Mass <- log(shorebird.data$M.Mass)
> shorebird.data$lgF.Mass <- log(shorebird.data$F.Mass)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
> crunchMod <- crunch(lgEgg.Mass ~ lgF.Mass + lgM.Mass, data=shorebird)
> par(mfrow=c(2,2))
> plot(crunchMod)
>
```

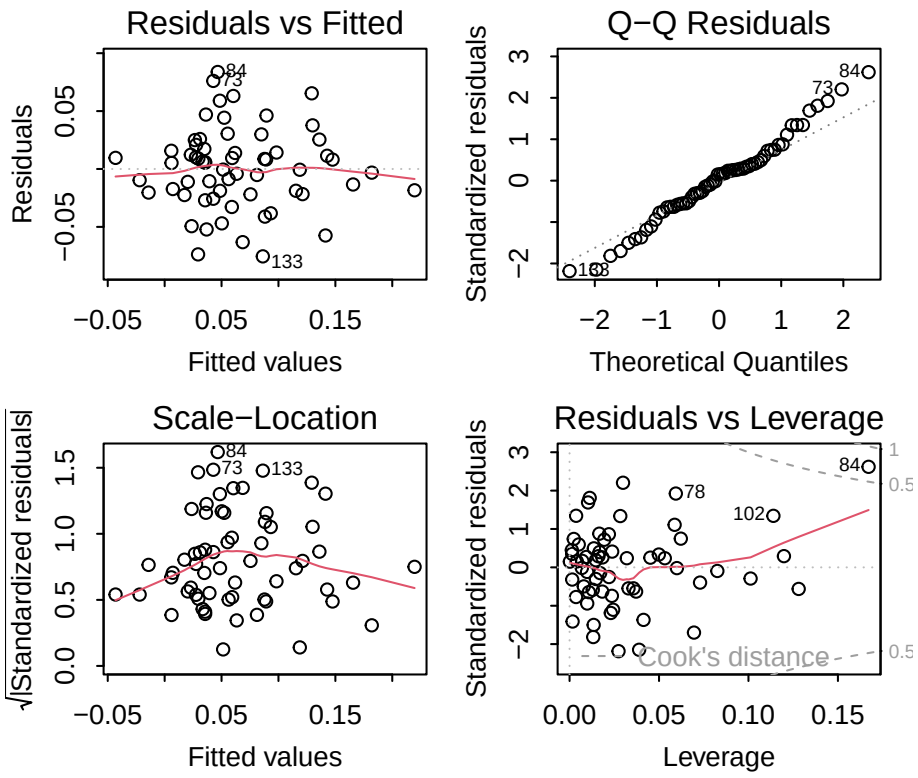


Figure 10: Testing the assumptions of a linear model for a contrast plot

3.3.4 Model comparison: `anova` & `AIC`

In a similar fashion, the `caper` package provides simple wrappers to standard `anova` and `AIC` methods to allow contrast models to be compared.

```
> shorebird.data$lgEgg.Mass <- log(shorebird.data$Egg.Mass)
> shorebird.data$lgM.Mass <- log(shorebird.data$M.Mass)
> shorebird.data$lgF.Mass <- log(shorebird.data$F.Mass)
> shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
> cMod1 <- crunch(lgEgg.Mass ~ lgM.Mass * lgF.Mass, data=shorebird)
> cMod2 <- crunch(lgEgg.Mass ~ lgM.Mass + lgF.Mass, data=shorebird)
> cMod3 <- crunch(lgEgg.Mass ~ lgM.Mass , data=shorebird)
> anova(cMod1, cMod2, cMod3)

Analysis of Variance Table

Model 1: lgEgg.Mass ~ lgM.Mass + lgF.Mass + lgM.Mass:lgF.Mass - 1
Model 2: lgEgg.Mass ~ lgM.Mass + lgF.Mass - 1
Model 3: lgEgg.Mass ~ lgM.Mass - 1
  Res.Df      RSS Df   Sum of Sq      F Pr(>F)
1     59 0.073257
2     60 0.073268 -1 -0.00001151 0.0093 0.9236
3     61 0.074363 -1 -0.00109437 0.8814 0.3516

> AIC(cMod1, cMod2, cMod3)

      df       AIC
cMod1  4 -233.9885
```

```
cMod2  3 -235.9787
cMod3  2 -237.0595
```

AIC

3.4 Other comparative functions

3.4.1 Tree imbalance: `fusco.test`

The I statistic (Fusco and Cronk, 1995) is a measure of phylogenetic imbalance. It has the useful properties that it can be calculated for trees containing polytomies and that the tips of trees can be associated with species richness values. An imbalance score, in the interval $[0, 1]$ is calculated for all nodes in the phylogeny and the distribution of the nodal values is used to assess overall balance. Nodes leading to fewer than 4 species are uninformative about balance and I cannot be calculated for polytomous nodes and hence both are omitted from balance calculations.

The original I statistic proposed by Fusco and Cronk (1995) has been shown to have a bias at nodes with even numbers of species (Purvis et al., 2002). This function therefore also implements the modified I' statistic: a Wilcoxon test is used to assess whether the median of the observed I' values differs from the value of 0.5 expected under a Markov process and a randomisation process is used to provide confidence intervals.

The function can make use of species richness data but can also be set to treat the tips of the phylogeny as species. These two approaches contrast the balance of the distribution of species richness across a topology and the balance of the topology itself.

```
> data(syrphidae)
> syrphidae <- comparative.data(phy=syrphidaeTree, dat=syrphidaeRich, names.col=genus)
> summary(fusco.test(syrphidae, rich=nSpp))

Fusco test for phylogenetic imbalance

  Tree with 105 informative nodes and 204 tips.
  Tips are higher taxa containing 5330 species.
  95.0% confidence intervals around 0.5 randomised using 1000 replicates.

  Mean I prime: 0.629 [0.434,0.567]
  Median I: 0.727
  Quartile deviation in I: 0.291

        Wilcoxon signed rank test with continuity correction

data:  object$observed$I.prime
V = 3986, p-value = 0.0001197
alternative hypothesis: true location is not equal to 0.5


> summary(fusco.test(syrphidae, tipsAsSpecies=TRUE))

Fusco test for phylogenetic imbalance

  Tree with 58 informative nodes and 204 tips.
  Tips are treated as species.
```

```
  95.0% confidence intervals around 0.5 randomised using 1000 replicates.

  Mean I prime: 0.714 [0.401,0.603]
  Median I: 0.94
  Quartile deviation in I: 0.25

        Wilcoxon signed rank test with continuity correction

data:  object$observed$I.prime
V = 1345.5, p-value = 0.0001481
alternative hypothesis: true location is not equal to 0.5
```

3.5 Phylogenetic diversity: `pd.calc`, `pd.bootstrap` and `ed.calc`.

There are two functions that provide simple ways to calculate phylogenetic diversity indices for a set of tips on a phylogeny (`pd.calc`) and to bootstrap the distribution of those indices for a subset of a given size (`pd.bootstrap`).

The following indices of phylogenetic diversity are supported (Fig. 11):

Total Branch Length (TBL) The sum of all the edge lengths in the subtree given by the tip subset. This measure can be partitioned into the two next measures.

Shared Branch Length (SBL) The sum of all edges in the subtree that are shared by more than one tip.

Unique Evolutionary History (UEH) The sum of the edge lengths that give rise to only one tip in the subtree.

Length of tip branch lengths (TIPS) Unlike UEH, this measure does not use the unique paths to each tips on the *subtree* and instead gives the sum of the unique branches leading to the tips on the *complete tree*.

Minimum Spanning Tree (MST) The sum of the lengths of the edges for the smallest tree that links the subset tips, excluding any edges below the node of the most recent common ancestor.

These two functions can be used together to compare observed phylogenetic diversity for a group of species to the distribution of phylogenetic diversity for a similar sized group. These functions use a phylogeny arranged as a `clade.matrix`, as this greatly accelerates sampling of phylogenetic diversity measures. A `phylo` object can be used but will be immediately converted to a clade matrix.

```
> data(BritishBirds)
> BritishBirds.cm <- clade.matrix(BritishBirds.tree)
> redListSpecies <- with(BritishBirds.data, binomial[Red_list==1])
> obs <- pd.calc(BritishBirds.cm, redListSpecies, method="TBL")
> rand <- pd.bootstrap(BritishBirds.cm, ntips=length(redListSpecies), rep=1000, method="TBL")
> plot(density(rand), main='Total branch length for random sets of 32 species.')
> abline(v=obs, col='red')
```

The `ed.calc` function takes a phylogeny, again ideally converted using `clade.matrix`, and calculates evolutionary distinctness (ED) scores for each tip (Isaac et al., 2007). The length of each branch on the phylogeny is shared equally among all the tips descending from it and the sum of the per-tip contributions along the paths to each tip provides the ED measure of the distinctness of each tip (Fig 12a). In the example code below, two species (*Allocebus trichotis*

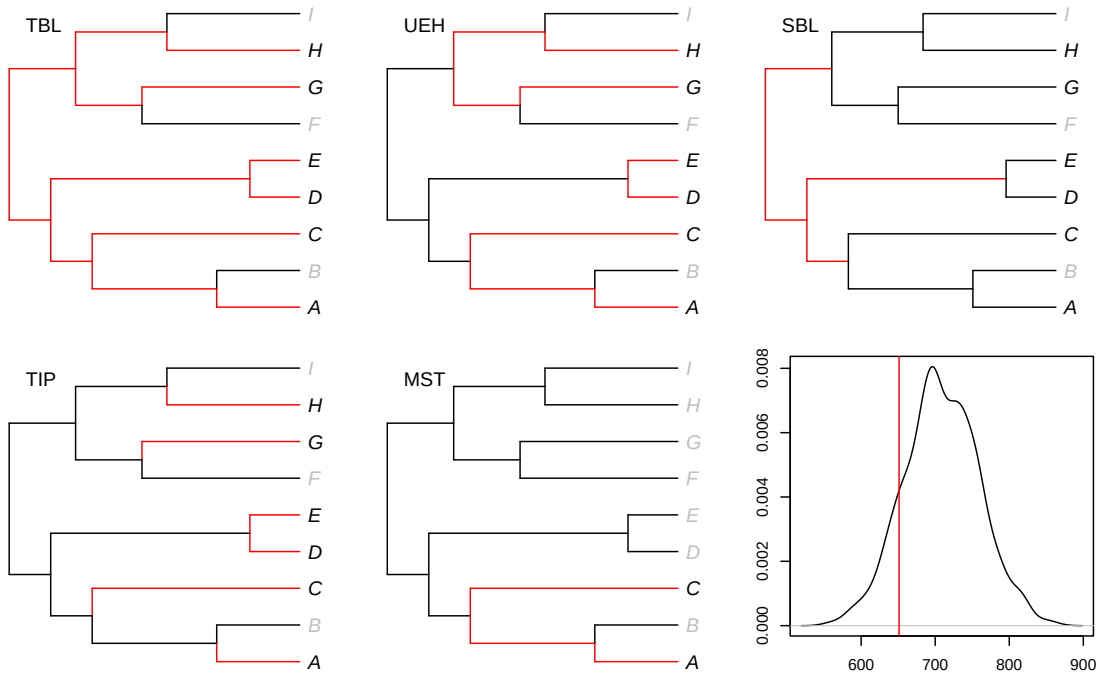


Figure 11: Illustration of the different measures of phylogenetic diversity available and a plot of observed (red) and randomized (black) TBL for red listed British birds

and *Phaner furcifer*) share long branches and belong to species poor clades, resulting in high evolutionary distinctiveness (Fig 12b,c)

```
> data(IsaacEtAl)
> primates.cm <- clade.matrix(primates.tree)
> primateED <- ed.calc(primates.cm)
> str(primateED)
```

```
List of 2
 $ spp   :'data.frame':        233 obs. of  2 variables:
  ..$ species: chr [1:233] "Lemur catta" "Hapalemur aureus" "Hapalemur griseus" "Hapalemur simus"
  ..$ ED     : num [1:233] 13.62 10.07 10.07 10.07 9.51 ...
 $ branch:'data.frame':        404 obs. of  6 variables:
  ..$ len      : num [1:404] 10.92 8.19 8.19 8.19 4.26 ...
  ..$ nSp      : num [1:404] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ ED       : num [1:404] 10.92 8.19 8.19 8.19 4.26 ...
  ..$ parent   : num [1:404] 240 241 241 241 244 244 245 245 242 238 ...
  ..$ node.size: int [1:404] 2 3 3 3 2 2 2 2 2 2 ...
  ..$ ED.cor   : num [1:404] 10.92 6.45 6.45 6.45 4.26 ...
```

```
> plot(density(primateED$spp$ED))
> with(primateED, spp[spp$ED == max(spp$ED),])
```

```
               species       ED
29 Allocebus trichotis 22.52876
30     Phaner furcifer 22.52876
```

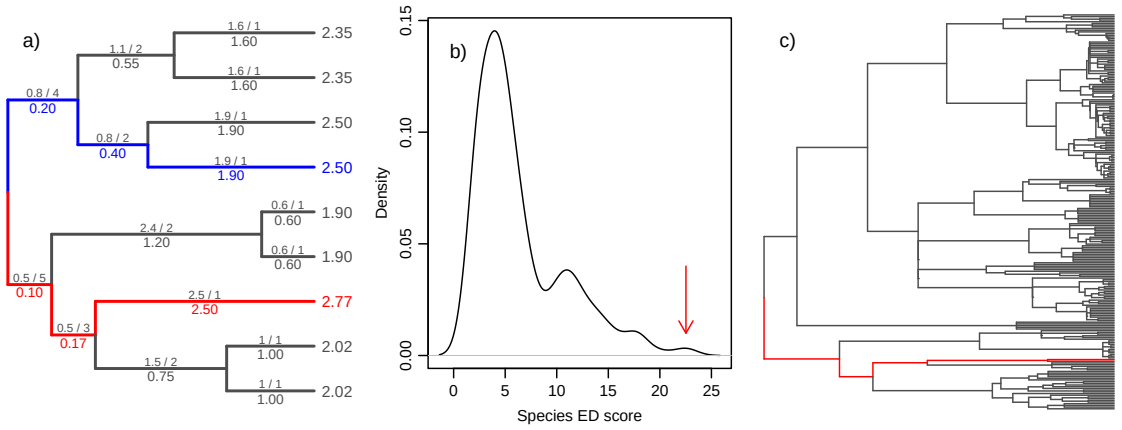


Figure 12: a) Calculation of ED scores. Edge labels show the calculation of ED contributions (below branch) from the branch length and number of descendent species (above branch). Tip labels show species ED scores and the component contribution values are highlighted for two species. b) The distribution of ED scores for a phylogeny of primate species (c). The paths leading to the species with the highest ED values (red arrow) are shown on the phylogeny.

3.6 Phylogeny and trait simulation: `growTree`

A number of other packages provide fast methods for simulating phylogenies and traits along phylogenies under a range of models. The `growTree` function is likely to be considerably slower for some common models but provides a flexible environment for simulating more complex scenarios. It bears many similarities to the simulations provided by the program MeSA written by Paul-Michael Agapow (http://www.agapow.net/software/mesa).

A basic `growTree` simulation has three components: a speciation rate (`b`), an extinction rate (`d`) and a stopping rule (`halt`). The default values for these (`b=1, d=0, halt=20`) therefore simply grow a phylogeny from a single ancestor until there are 20 extant species. The speciation and extinction rates both specify the rate of an exponential function which is used to draw waiting times until the next event.

However, the power of `growTree` is that any of these components can be R expressions using information about the scenario being simulated. In simple simulations of a phylogeny, the following clade properties can be used: `clade.age`,`nLin`, `nTip`,`nExtantTip`, `nExtinctTip`. For example, the default `halt = 20` is actually just a shorthand for `expression(nExtantTip == 20)` and more examples are shown in the code below and in Fig. 13.

There are three important details of these simple simulations:

1. Expressions using `clade.age` *cannot* test for an exact end time (`clade.age==3`) since the simulation may overstep this end point and run infinitely: `clade.age>=3` must be used instead.
2. Expressions using numbers of lineages or tips will halt the simulation exactly at a speciation or extinction event. The optional argument `extend.proportion` can be used to continue to run the simulation until a given proportion of the time to the next speciation has passed.
3. The `clade.age` includes the length of the root edge.

The significance of the argument `grain=Inf` in these examples is discussed below (Section 3.6.3).

```
> basicTree <- growTree(b=1, d=0, halt=20, grain=Inf)
> extendTree <- growTree(b=1, d=0, halt=20, extend.proportion=1, grain=Inf)
```

```
> timeLimit <- growTree(b=1, d=0, halt=expression(clade.age>=3), grain=0.01)
> extinctTree <- growTree(b=1, d=0.1, halt=expression(nExtinctTip>=3), extend.proportion=1, grain=
> densityDependence <- growTree(b=expression(2/nExtantTip), d=0, halt=50, grain=Inf)
> increasingExtinction <- growTree(b=1, d=expression(0.2 * clade.age), halt=50)
```

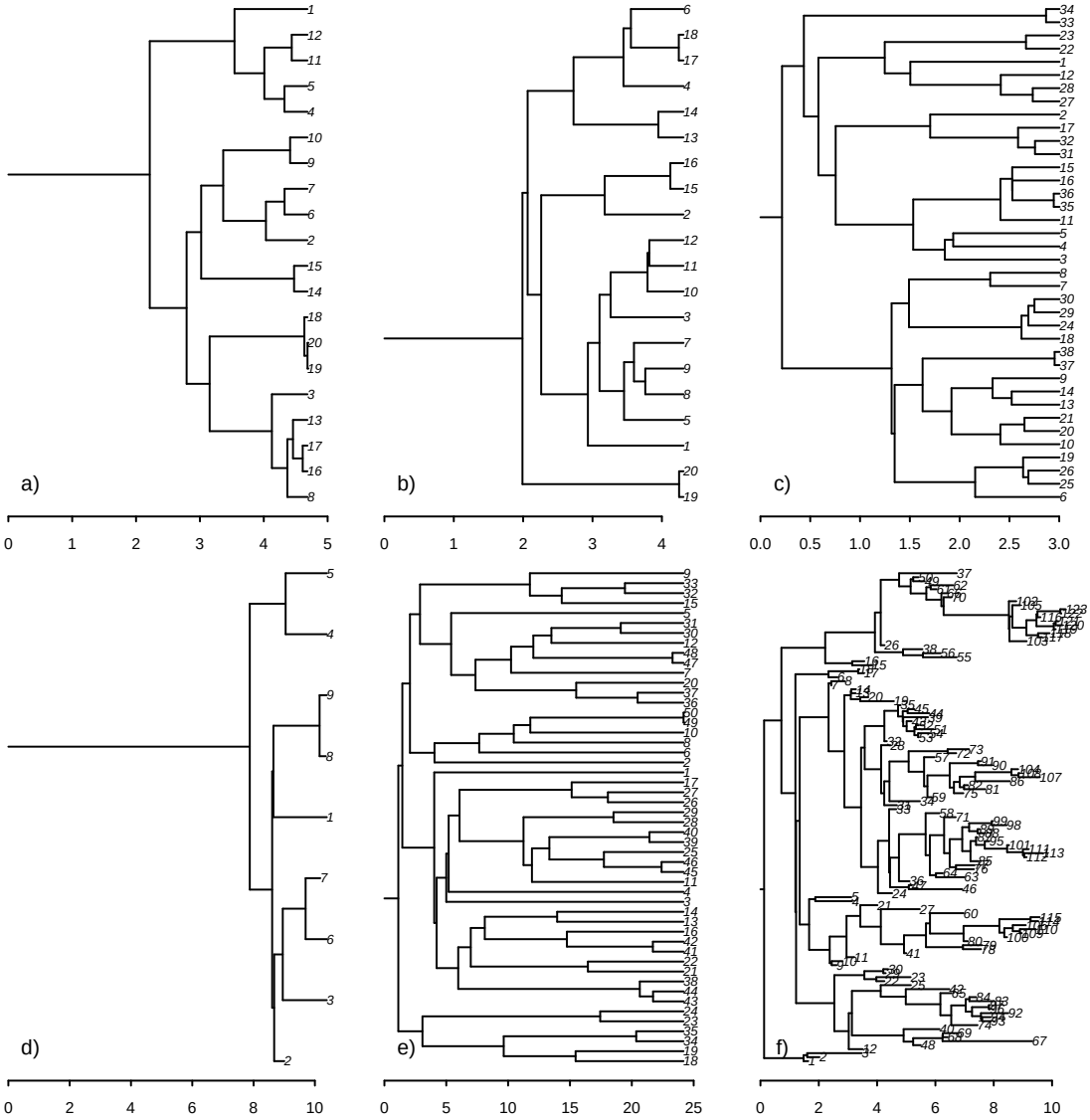


Figure 13: growTree simulations
Simple simulations using `growTree`. See the code example for a) `basicTree`, b) `extendTree`, c) `timeLimit`, d) `extinctTree`, e) `densityDependence` and f) `increasingExtinction`.

3.6.1 Simulating continuous characters.

The `growTree` function currently only supports Brownian evolution of continuous characters and requires the user to provide a vector containing of starting value for each trait (`ct.start`). If `ct.start` is a named vector, then these names will be used to identify each trait within the simulation. If no names are provided then the trait names default to the form `ct#`.

By default, the mean change per unit time in each trait is assumed to be zero — there is no directional bias in the evolution of each trait. Similarly, the variance of each trait is assumed to be 1 and there is no covariance between traits. Both these defaults can be over-ridden by providing a

vector `ct.change` of mean changes and a vector `ct.var` of variances for each trait. Alternatively, `ct.var` can be a symmetrical square matrix giving the covariances between the traits.

During the simulation, the waiting times between events are used to randomly draw changes in trait values according to the specified model. The code below shows some examples using values for log body mass and log litter size taken from the PanTHERIA database of mammalian life-history traits (Jones et al., 2009). The relationship between the two variables for the values at the tips of each simulation are shown in Fig. 14.

```
> # > pantheria <- read.delim('PanTHERIA_1-0_WR05_Aug2008.txt', na.string='-999.00')
> # > vars <- log(pantheria[, c(7,21)])
> # > cov(vars, use='complete')
> #                       X5.1_AdultBodyMass_g X15.1_LitterSize
> # X5.1_AdultBodyMass_g           10.0038050       -0.6544125
> # X15.1_LitterSize               -0.6544125        0.4427426
> # > sum(complete.cases(vars))
> # [1] 2325
> # > mean(vars,na.rm=TRUE)
> # X5.1_AdultBodyMass_g     X15.1_LitterSize
> #            5.4749827            0.6905053
> mammalMeans <- c(logBodyMass=5.48, logLitterSize=0.69)
> simpleBrownian <- growTree(halt=100, ct.start=mammalMeans, extend.proportion=1, grain=Inf)
> mammalVar <- c(10, 0.44)
> varBrownian <- growTree(halt=100, ct.start=mammalMeans, ct.var=mammalVar, extend.proportion=1, g
> mammalCovar <- matrix(c(10,-0.65, -0.65, 0.44), ncol=2)
> covarBrownian <- growTree(halt=100, ct.start=mammalMeans, ct.var=mammalCovar, extend.proportion=
```

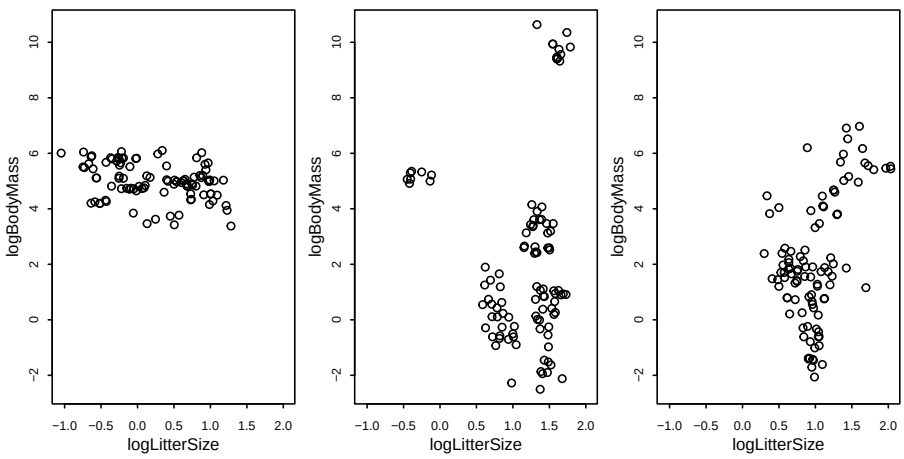


Figure 14: Simulated values for two continuous traits using a) only starting mean values (`simpleBrownian`), b) means and variances (`varBrownian`) and c) means and covariances (`covarBrownian`).

3.6.2 Simulating discrete characters

Simulation of discrete characters uses a transition matrix for each trait. This matrix defines the number of states and the names of those states and also the rates at which lineages move from one state to another. This matrix does not need to be symmetrical but the diagonal values (the rates at which traits change to the same state) should be zero. These discrete trait matrices need to be provided in a list and the names of the list items will be used as trait names. Again, if names are missing than default values are used: `dt#` for trait names and `st#` for state names. The simulation will start with the traits in the first state in the matrix.

```
> flight <- matrix(c(0,0.1,0.001,0), ncol=2)
> flightNames <- c('ter','fly')
> dimnames(flight) <- list(flightNames, flightNames)
> trophic <- matrix(c(0,0.2,0.01,0.2,0,0.1,0,0.05,0), ncol=3)
> trophNames <- c('herb','omni','carn')
> dimnames(trophic) <- list(trophNames, trophNames)
> discTraits <- list(flight=flight, trophic=trophic)
> discreteTree <- growTree(halt=60, dt=discTraits, extend.proportion=1, grain=Inf)
>
```

Simulating correlated evolution between discrete traits is possible by combining traits into a single larger matrix that contains the rates of transitions between sets of combined states.

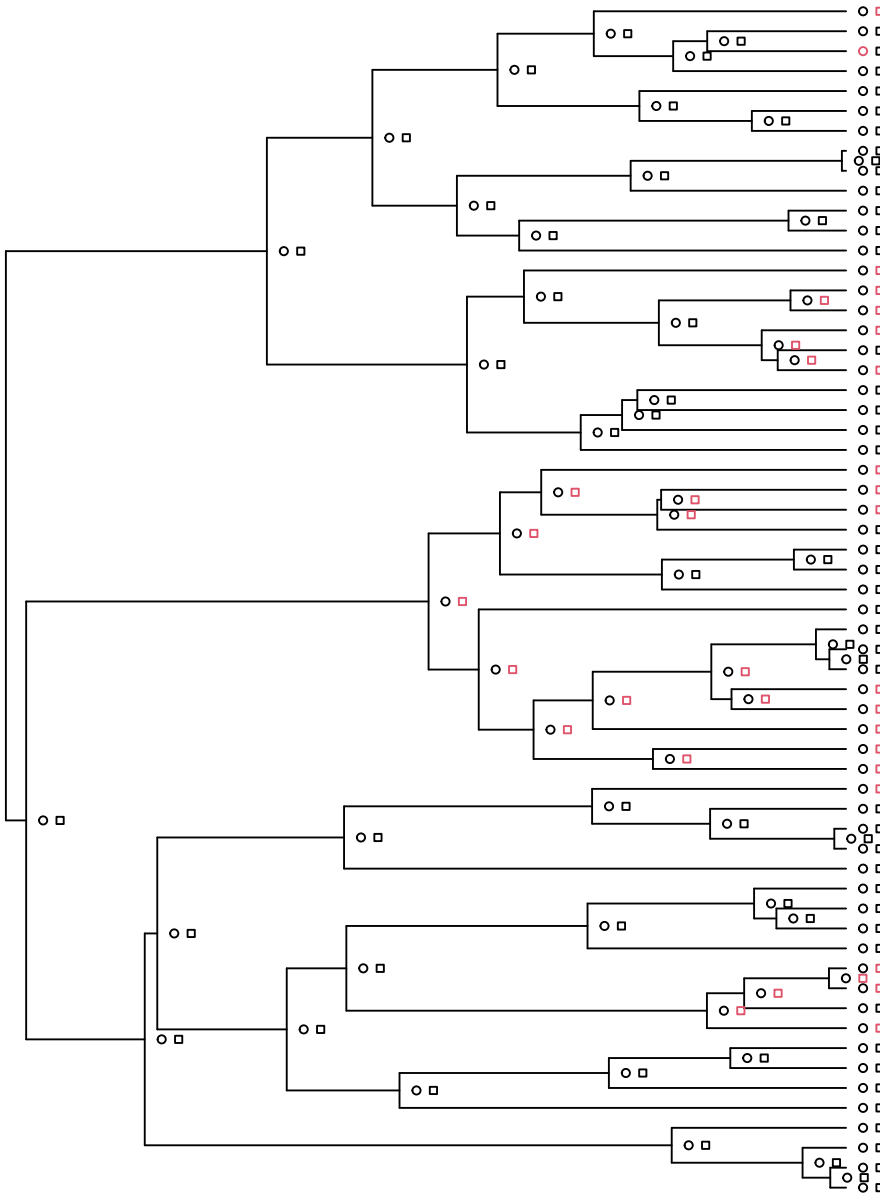


Figure 15: Simulated values for two discrete traits: `flight` (circles) and `trophic` (squares).

3.6.3 Lineage properties and simulation grain

In addition to the clade properties (page 24), expressions for speciation and extinction rates and halting the simulation can all use properties of the lineages. The basic lineage properties are less likely to be useful (`parent.id, id, lin.age, birth.time, death.time, extinct, tip`) but any simulated trait is also available using the trait name. Rates are calculated for every extant lineage and then the competing events (speciation, extinction and changes in discrete traits) draw waiting times given their rates and the event with the shortest waiting time occurs.

These events occur at the discrete points identified by the winning event. However, change in continuous traits is handled differently: once one of those events has occurred, that waiting time is used to scale simulated change in the continuous traits and all of this change is added on to the trait values at the event. Similarly, clade age advances in chunks: the waiting time is added on to the age of all extant tips as each event occurs.

If continuous trait values or clade age are used to control rates within the simulation then this can be problematic: smooth changes in rates may be badly simulated if there are relatively large step changes in continuous character values or clade age at events. The solution in `growTree` is to set the `grain` of the simulation: this is the maximum amount of time that can pass in a simulation without recalculating clade ages and continuous trait values. Essentially, this scales the relative size of step changes in these continuous values to more closely approximate a smooth function.

By default, `growTree` uses a grain value of 0.1. If a particular simulation does not contain rates base on trait or age values then the grain can be set to infinity (`grain=Inf`). This may substantially speed up some simulations.

In the examples below, the age of each lineage is being used to control the rates of speciation and extinction (Fig. 16). Speciation follows an asymptotic latency model: young species have lower rates of speciation but eventually reach a constant maximum speciation rate. Extinction follows a simple linear relationship of senescence: increasing extinction rate with lineage age. The second simulation incorporates an influence of body size: the asymptotic speciation rate for species is proportional to a simulated trait.

```
> bLatency <- expression((0.2 *lin.age)/(0.1 + lin.age))
> dSenesce <- expression(lin.age * 0.02)
> halt <- expression(nExtantTip >= 60)
> latencyTree <- growTree(b=bLatency, d=dSenesce, halt=halt, grain=0.01)
> traitMean <- c(logBodyMass=5.48)
> traitVar <- c(logBodyMass=10)
> bLatTrait <- expression(((0.2 + ((5.48 - logBodyMass)/25))*lin.age)/(0.1 + lin.age))
> latTraitTree <- growTree(b=bLatTrait, d=dSenesce, halt=halt, ct.start=traitMean, ct.var=traitVa
>
```

3.6.4 Inheritance rules.

Another way of controlling simulations is through the use of inheritance rules. At each speciation event, two new lineages replace the parent lineage and inherit exact copies of any traits. Although the traits of these new species will then evolve independently, it is also possible to immediately alter the trait values using expressions. Each rule should generate two values, one for each daughter, and needs to be provided as a list with the name of the list showing the trait affected.

In the first example below, the two daughter lineages inherit divergent body size from their parent (Fig. 17).

```
> traitMean <- c(logBodyMass=5.48)
> traitVar <- c(logBodyMass=10)
```

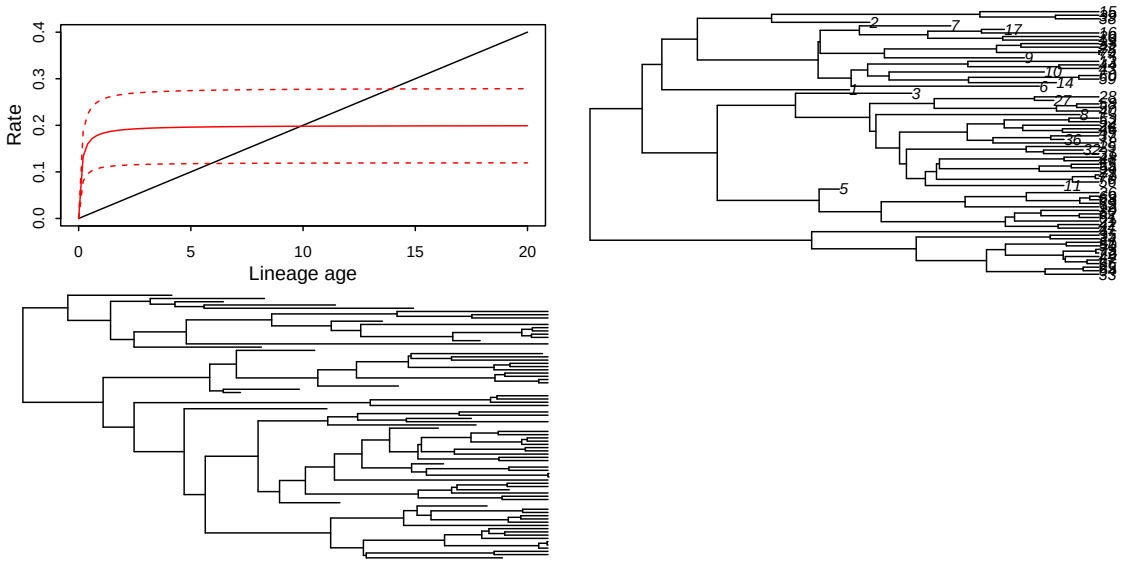


Figure 16: Simulations using expressions. a) Simulated extinction and speciation rates. Extinction (black) shows senescence and speciation shows latency. For the first simulation (b: `latencyTree`), speciation is determined entirely by lineage age and there are relatively few extant basal species and few rapid sequences of speciation along branches. For the second (c: `latTraitTree`), there are again few rapid sequence of speciation and simulated body mass (red bars) is predominantly smaller than the ancestral average (width of the grey bar). The dashed red lines in a) show the speciation rate for species with simulated log body mass 2 units higher and lower than the ancestral mean.

```
> inherit <- list(logBodyMass=expression(logBodyMass * c(0.8, 1.2)))
> inheritTree <- growTree(b=1, halt=40, ct.start=traitMean, ct.var=traitVar, inheritance=inherit,
```

The second example is a more complex biogeographic simulation (Fig. 17). The tree has a single discrete trait representing presence in two regions — a species can be present in either or both regions and the matrix shows the rates of transition between these states. The extinction rule gives different extinction rates for species occurring in single regions and species in both regions. Speciation occurs at different rates in the two regions and species present in both regions can speciate in either region. This needs two speciation rules, provided as a list, that compete with each other and among species. Finally the inheritance rules differentiate between speciation of a species in a single region (where both species stay in sympatry) and speciation in a lineage that occurs in both regions (where the species diverge in allopatry). These inheritance rules make use of the simulation property `winnerName` which shows which event has happened at an event.

```
> rNames <- c("AB", "A", "B")
> regions <- list(region = matrix(c(0, 0.05, 0.05, 0.1, 0,0,0.1,0,0), ncol=3,
+                       dimnames=list(rNames,rNames)))
> extinct <- expression(ifelse(region == "AB", 0.0001, 0.01))
> spec <- list(regA_spec = expression(ifelse(region == "AB" | region == "A", 1, 0)),
+              regB_spec = expression(ifelse(region == "AB" | region == "B", 0.5, 0)))
> inherit <- list(region = expression(if(winnerName=="regB_spec" && region[1] == "AB") c("B","A")
+                 region = expression(if(winnerName=="regA_spec" && region[1] == "AB") c("A","B")
> biogeogTree <- growTree(b=spec, d=extinct, halt=80, inherit=inherit, dt.rates=regions, inf.rates
```

3.6.5 Epochs: linking simulations.

The final way of controlling simulations is through the use of linked sets of simulations, each representing a different epoch. By default, the `growTree` function returns a comparative data

object but can also return a simulation as a raw table of lineages. These lineage tables can be used as a starting object for a simulation, allowing the output of one simulation to be passed into the next. Unless explicitly altered, all rules are inherited between epochs. The example below simulates a mass extinction event (Fig. 17).

```
> epochTree <- growTree(halt=100, output.lineages=TRUE, grain=Inf)
> epochTree2 <- growTree(d=5, halt=expression(nExtantTip==20), linObj=epochTree, output.lineages=T
> epochTree3 <- growTree(linObj=epochTree2, halt=100, grain=Inf)
```

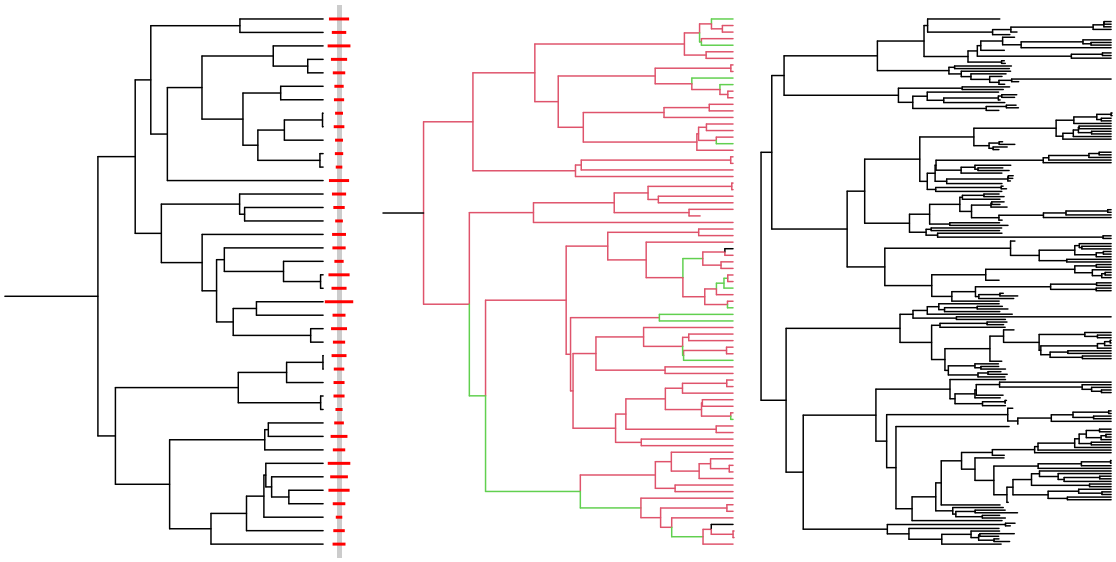


Figure 17: Inheritance rules and epochs. a) Simulation showing divergent evolution of a continuous trait at speciation. b) A biogeographic simulation with sympatric speciation within single regions (red, green) and allopatric speciation of species occurring in both regions (black). Colours show the region at daughter node of an edge and do not show transitions along the edge. c) Use of epochs to simulate a mass extinction event.

3.7 Utility functions

3.7.1 clade.matrix

The clade.matrix function turns a phylogeny into a binary matrix showing the tips (columns) descending from each node (rows) in the tree, along with a vector of the edge lengths leading to that node. Although this is an extremely bulky way to represent a phylogeny, a clade matrix allows calculations on subsets of a phylogeny to be calculated much more rapidly.

```
> data(perissodactyla)
> perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial)
> perissoCM <- clade.matrix(perisso$phy)
> str(perissoCM)
```

```
List of 4
 $ clade.matrix: num [1:16, 1:9] 1 0 0 0 0 0 0 0 0 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ edges: chr [1:16] "1" "2" "3" "4" ...
  .. ..$ tips : chr [1:9] "1" "2" "3" "4" ...
 $ tip.label   : chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" "Tapiru
 $ edge        : int [1:15, 1:2] 10 11 12 12 13 13 11 10 14 15 ...
 $ edge.length : Named num [1:16] 4 1 1 8 1 1 4 8 8 0 ...
  ..- attr(*, "names")= chr [1:16] "1" "2" "3" "4" ...
 - attr(*, "class")= chr "clade.matrix"
```

```
> perissoCM$clade.matrix
```

```
     tips
edges 1 2 3 4 5 6 7 8 9
   1  1 0 0 0 0 0 0 0 0
   2  0 1 0 0 0 0 0 0 0
   3  0 0 1 0 0 0 0 0 0
   4  0 0 0 1 0 0 0 0 0
   5  0 0 0 0 1 0 0 0 0
   6  0 0 0 0 0 1 0 0 0
   7  0 0 0 0 0 0 1 0 0
   8  0 0 0 0 0 0 0 1 0
   9  0 0 0 0 0 0 0 0 1
   10 1 1 1 1 1 1 1 1 1
   11 1 1 1 1 0 0 0 0 0
   12 1 1 1 0 0 0 0 0 0
   13 0 1 1 0 0 0 0 0 0
   14 0 0 0 0 1 1 1 1 1
   15 0 0 0 0 1 1 1 0 0
   16 0 0 0 0 1 1 0 0 0
```

3.7.2 clade.members and clade.members.list

These functions return a vector of the tips that descend from a given node (clade.members) or a list of vectors showing the tips descending from every node (clade.members.list) in a phylogeny. Options to both functions control whether vectors of internal nodes are also returned (include.nodes) and whether tip labels are used instead of the index of those tip labels (tip.labels).

```
> data(perissodactyla)
> perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial)
> clade.members(15, perisso$phy)

[1] 7 5 6

> clade.members(15, perisso$phy, tip.labels=TRUE)

[1] "Equus zebra"     "Equus grevyi"    "Equus burchelli"

> clade.members(15, perisso$phy, tip.labels=TRUE, include.nodes=TRUE)

$tips
[1] "Equus zebra"     "Equus grevyi"    "Equus burchelli"

$nodes
[1] 15 16

> str(clade.members.list(perisso$phy))

List of 7
 $ 10: int [1:9] 4 1 2 3 8 9 7 5 6
 $ 11: int [1:4] 4 1 2 3
 $ 12: int [1:3] 1 2 3
 $ 13: int [1:2] 2 3
 $ 14: int [1:5] 8 9 7 5 6
 $ 15: int [1:3] 7 5 6
 $ 16: int [1:2] 5 6

> str(clade.members.list(perisso$phy, tip.labels=TRUE))

List of 7
 $ 10: chr [1:9] "Tapirus indicus" "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum"
 $ 11: chr [1:4] "Tapirus indicus" "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum"
 $ 12: chr [1:3] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum"
 $ 13: chr [1:2] "Diceros bicornis" "Ceratotherium simum"
 $ 14: chr [1:5] "Equus africanus" "Equus hemionus" "Equus zebra" "Equus grevyi" ...
 $ 15: chr [1:3] "Equus zebra" "Equus grevyi" "Equus burchelli"
 $ 16: chr [1:2] "Equus grevyi" "Equus burchelli"
```

3.7.3 VCV.array

This function replaces the `vcv.phylo` function provided by the ape package. This is primarily to provide covariance structures that retain separate branch lengths in order to support kappa transformations. For standard two dimensional covariance matrices, it is also faster than the `vcv.phylo` function for trees of more than about 100 tips.

```
> data(perissodactyla)
> perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial)
> str(VCV.array(perisso$phy))
```

```
 'VCV.array' num [1:9, 1:9] 17 13 13 9 0 0 0 0 0 13 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" "Tapirus indicus
  ..$ : chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" "Tapirus indicus
```

```
> str(VCV.array(perisso$phy, dim=3))
```

```
 'VCV.array' num [1:9, 1:9, 1:5] 4 9 9 9 0 0 0 0 0 9 ...
 - attr(*, "dimnames")=List of 3
  ..$ : chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" "Tapirus indicus
  ..$ : chr [1:9] "Rhinoceros unicornis" "Diceros bicornis" "Ceratotherium simum" "Tapirus indicus
  ..$ : NULL
```

References

Paul-Michael Agapow and Nick J B Isaac. Macrocaic: revealing correlates of species richness by comparative analysis. *Diversity and Distributions*, 8:41–43, 2002.

Austin Burt. Comparative methods using phylogenetically independent contrasts. *Oxford Surveys in Evolutionary Biology*, 6:33–53, 1989.

Joseph Felsenstein. Phylogenies and the comparative method. *American Naturalist*, 125:1–15, 1985.

Robert P. Freckleton, Paul H. Harvey, and Mark Pagel. Phylogenetic analysis and comparative data: A test and review of evidence. *American Naturalist*, 160:712–726, 2002.

Susanne A. Fritz and Andy Purvis. Selectivity in mammalian extinction risk and threat types: a new measure of phylogenetic signal strength in binary traits. *Conservation Biology*, 24(4): 1042–1051, August 2010. doi: DOI 10.1111/j.1523-1739.2010.01455.x.

Guiseppe Fusco and Quentin C B Cronk. A new method for evaluating the shape of large phylogenies. *Journal of Theoretical Biology*, 175:235–243, 1995.

Theodore Garland, Paul H. Harvey, and Anthony R. Ives. Procedures for the analysis of comparative data using phylogenetically independent contrasts. *Systematic Biology*, 41(1):18–32, Mar 1992. ISSN 1063-5157.

Nick J. B. Isaac, Samuel T. Turvey, Ben Collen, Carly Waterman, and Jonathan E. M. Baillie. Mammals on the edge: Conservation priorities based on threat and phylogeny. *Plos One*, 2(3): e296, March 2007. doi: DOI 10.1371/journal.pone.0000296.

NJB Isaac, PM Agapow, PH Harvey, and A Purvis. Phylogenetically nested comparisons for testing correlates of species richness: A simulation study of continuous variables. *Evolution*, 57 (1):18–26, January 2003.

NJB Isaac, KE Jones, JL Gittleman, and A Purvis. Correlates of species richness in mammals: Body size, life history, and ecology. *American Naturalist*, 165(5):600–607, May 2005.

Kate E. Jones and Andy Purvis. An optimum body size for mammals? comparative evidence from bats. *Functional Ecology*, 11(6):751–756, 1997. ISSN 1365-2435. doi: 10.1046/j.1365-2435.1997.00149.x. URL `http://dx.doi.org/10.1046/j.1365-2435.1997.00149.x`.

Kate E. Jones, Jon Bielby, Marcel Cardillo, Susanne A. Fritz, Justin O'Dell, C. David L. Orme, Kamran Safi, Wes Sechrest, Elizabeth H. Boakes, Chris Carbone, Christina Connolly, Michael J. Cutts, Janine K. Foster, Richard Grenyer, Michael Habib, Christopher A. Plaster, Samantha A. Price, Elizabeth A. Rigby, Janna Rist, Amber Teacher, Olaf R. P. Bininda-Emonds, John L. Gittleman, Georgina M. Mace, Andy Purvis, and W. K. Michener. Pantheria: a species-level database of life history, ecology, and geography of extant and recently extinct mammals. *Ecology*, 90(9):2648–2648, 2009. doi: 10.1890/08-1494.1. URL `http://www.esajournals.org/doi/abs/10.1890/08-1494.1`.

A Katzourakis, A Purvis, S Azmeh, G Rotheray, and F Gilbert. Macroevolution of hoverflies (diptera : Syrphidae): the effect of using higher-level taxa in studies of biodiversity, and correlates of species richness. *Journal of Evolutionary Biology*, 14:219–227, 2001.

Terje Lislevand and Gavin H. Thomas. Limited male incubation ability and the evolution of egg size in shorebirds. *Biology Letters*, 2(2):206–208, June 2006. doi: DOI 10.1098/rsbl.2005.0428.

Mark D Pagel. A method for the analysis of comparative data. *Journal of Theoretical Biology*, 156 (4):431–442, 1992.

Emmanuel Paradis, Julien Claude, and Korbinian Strimmer. Ape: analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20(2):289–290, 2004.

Andy Purvis and Andrew Rambaut. *Comparative Analysis by Independent Contrasts (CAIC) User's Guide*, 1995a.

Andy Purvis and Andy Rambaut. Comparative analysis by independent contrasts (caic): an apple macintosh application for analysing comparative data. *Computer Applications In the Biosciences*, 11:247–251, 1995b.

Andy Purvis, Aris Katzourakis, and Paul-Michael Agapow. Evaluating phylogenetic tree shape: Two modifications to fusco & cronk's method. *Journal of Theoretical Biology*, 214:99–103, 2002. doi: DOI 10.1006/jtbi.2001.2443.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

Gavin H. Thomas. Phylogenetic distributions of british birds of conservation concern. *Proceedings of the Royal Society B-Biological Sciences*, 275(1647):2077–2083, September 2008. doi: DOI 10.1098/rspb.2008.0549.

Index

A The `CAIC` 'package'

The `CAIC` package was initially developed and used within the Section of Ecology and Evolution at Imperial College London as a replacement for the `CAIC` program (Purvis and Rambaut, 1995b). Over time, the `CAIC` package was circulated outside of the Section and accreted additional functions for comparative analysis and was subsequently made available through the R-Forge website. However, there was little integration of the different data formats across functions and no consistent provision of basic methods and documentation to support the main functions.

The `caper` package addresses these failings and replaces the package `CAIC`. Although the code and functions largely mirror the `CAIC` package, many functions are now used in different ways – notably through the use of a common comparative data structure – and function arguments have changed. Where possible these changes are backwards compatible but the two packages are not completely interchangeable. One of the more noticeable changes is that the function `pglm` has been renamed to `pgls`: the function provides generalised least squares but does not provide the link functions of generalised linear models. The change in name of the package therefore reflects both the broader set of methods provided and this step change in the way the package is used.