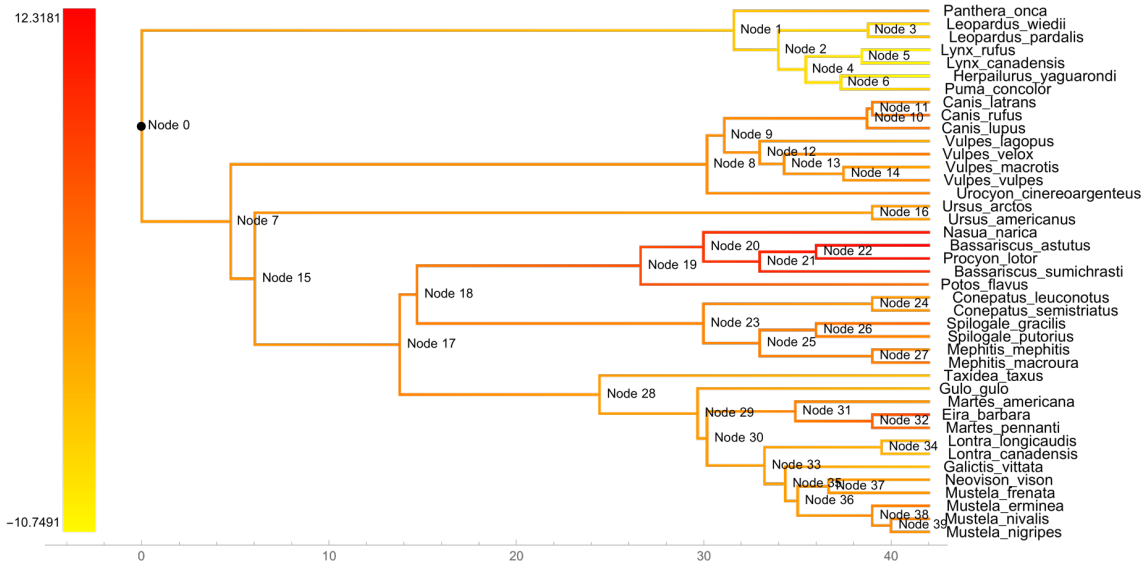
Phylogenetics for Mathematica

User's Guide

Version 6.8, June 2023



P. David Polly
Department of Earth & Atmospheric Sciences
Indiana University
Bloomington, Indiana 47405 USA

https://pollylab.indiana.edu/
pdpolly@indiana.edu

Cite as:

Polly, P.D. 2023. Phylogenetics for Mathematica. Version 6.8.
https://github.com/pdpolly/Phylogenetics-for-Mathematica

Table of Contents

Change list

6.8 Updated ***PagelLambda*** and ***PhylogeneticRegressionWithLambda*** to use PseudoInverse (i.e., Moore/Penrose inverse) to avoid errors with singular matrices when lambda is large.

6.7 ***PhylogeneticRegressionWithLambda*** **(**new function): Allows users to scale the internal branches of a phylogenetic tree with a Pagel's lambda value to conform to a non-Brownian motion model of evolution. ***UltrametricizeTree*** (new function): Turns a non-ultrametric tree into ultrametric by lengthening all tip branches to be coeval with the longest tip. ***DrawNewickTree*** (improved function): added new autoscaling capability and user-set aspect ratio option to the graphical output of *DrawNewickTree*.

6.6 Rewrote ***PagelsLambda*** and added ***LambdaMultiplication***

6.5. ***PhylogeneticRegression:*** Corrected error in mean squares, F-ratio, and p-value in

6.4. ***CtoTree*** (new function): Converts a phylogenetic covariance matrix into a Newick tree.

6.3 Changed Modules to Blocks to better localize variables used in functions.

6.2 ***MapTraitsOntoTree*** (improved function): Color scaling now differentiates between values that have decreased and increased from estimated ancestral trait value and there is now an option to show node labels.

PhylogeneticRegression **(improved function):** This function now returns an ANOVA table, R-squared value, SE of the regression and the predicted trait values.

6.1 ***MapTraitsOntoTree*** **(improved function):** Added options to allow user to scale max and min trait values.

DrawNewickTree **and** ***MapTraitsOntoTree*** **(improved functions):** Changed formatting of tip labels to make the trees easier to import into Adobe Illustrator and other vector-based editing packages.

PruneTree **(improved function):** Fixed bug that caused function to fail on some large trees.

6.0 ***PhylogeneticRegression*** **(new function)**: Performs a PGLS regression using the method of Martins and Hansen (1997).

PruneTree **(new function):** Selects a new tree based on a subset of taxa from a larger tree.

MapTraitsOntoTree **(new function)**: Draws a color-coded tree using reconstructed trait values.

OMearaAncestor **(new function)**: Reconstructs ancestral node value for base of a tree.

OMearaRate **(new function):** Estimates evolutionary rate (or rate matrix) from trait data.

ReconstructNodesSimple (new function): Reconstructs trait values for all nodes in a tree.

Kappa **(new function):** Returns Blomberg's *K* (kappa), an index the describes the proportion of trait variance attributable to phylogenetic covariance.

KappaMultivariate **(new function)**: Returns Adams' multivariate *K* (kappa), an extension of Blomberg's *K* for highly multivariate data such as geometric morphometric data.

***SharedPhylogeneticHistory* (new function):** Returns statistics of closeness of relationship of a subset of taxa from a larger clade, particularly for use in analysis of phylogenetic community structure.

MeanTreeDistance (new function): Returns mean patristic distance among subset of taxa on a larger tree, analogous to Webb's mean node distance used in analysis of phylogenetic community assembly.

***DrawNewickTree* (improved function):** Added error-trapping features.

5.1 ***ThreeModelTest (improved function):*** Updated the LogLikeRandWalk[] and LogLikeStasis[] sub functions to remove minor inconsistencies with Hunt (2006).

5.0 ***ThreeModelTest (new function*):** Performs evolutionary model selection on trait data from an evolving lineage as described by Hunt (2006).

PagelsLambda (new function): Calculates Pagel's Lambda for a univariate continuous trait on a phylogenetic tree.

4.1 ***ReconstructNodes (improved function)*:** Fixed a bug in the code that determines the number of dimensions to use in the node reconstruction that caused failure on datasets in which the number of taxa was much lower than number of landmark dimensions.

4.0 ***ReadNewick (improved function):*** Newick format files without branch length specifications are now automatically imported with unit branch lengths.

ReconstructNodes (improved function): fixed incompatibilities that caused problems with ***TreeToMorphospace[]*** and ***PhylogeneticPrincipalComponents[]*** functions in the Morphometrics for Mathematica package.

2.3 ***LineageEvolution (new function)***: Models trait evolution as Brownian Motion, OU, or Directional process (or combination thereof).

Installing the package

1. Download the latest version of the package at https://github.com/pdpolly/Phylogenetics-for-Mathematica (copy and paste into a text file and save with extension .m)
2. Open the file in Mathematica
3. Under the File menu, choose "Install"
4. Under Type of Item choose "Package", under source choose the file you just saved, under Install Name choose a short name for the package (e.g., "PollyPhylogenetics")
5. Once installed, enter the command "<<PollyPhylogenetics`" to use the functions.
6. Use the function *PhylogeneticsVersion[]* to determine which version you have installed.

Using Mathematica

Mathematica has a unique interface that takes a while to get used to. You open to a blank page, like a word processor, where you can type anything you want. Most of the

time you will type commands that do things with your data: connect to databases, plot graphs, carry out calculations. Unlike other statistical or mathematical programs, the commands you type and the output you get remain on the page, which gives you a record of what you've done step-by-step. To help organize your work, you can add headers, format boxes, etc. with the Format Menu.

Cells are an important organizing feature of *Mathematica*. Note the "cell" markers on the right margin. Each cell is bounded by a bracket and commands within a cell are executed together. You can open and close cells by double clicking the bracket. This can be useful if you have lots of stuff in a notebook… you can give a section a heading, which causes cells in that section to be grouped, after which you can close the section by double clicking.

Shift + Enter causes a cell to be executed. Pressing the enter key creates a new line, just like in a word processor, but when you want to execute a command you typed, you type SHIFT+ENTER somewhere in the cell and all commands in the cell are executed.

Mathematica Commands. Mathematic is designed to be as easy to learn as possible so that you can concentrate on working instead of the program. Almost all commands are English words written out in full with capitals at the beginning of words and brackets [] at the end of the command. For example, the command to calculate an average of a set of numbers is *Mean[]*, the command to do a principal components analysis is *PrincipalComponents[]*, and the command to take the logarithm of a number is *Log[]*.

Formatting Output. Mathematica is clever about how it provides output and it tries to keep the results as accurate as possible. For example, if you calculate the average of the following numbers

```
Mean[ {1, 5, 10, 3, 20, 40} ]
```

the answer extends to many decimal places, so Mathematica reports it more precisely as a fraction: 79/6. You may want an ordinary number, however, and you can force Mathematica to format its output the way you want:

```
Mean[ {1, 5, 10, 3, 20, 40} ] // N
```

This command now gives you 13.1667. Another useful formatting function is **//MatrixForm**, which causes a table to be displayed neatly in columns instead of wrapping around the page.

Graphics. Mathematic is good at graphics. You can either use simple functions like *ListPlot[]* to create a generic graph, or you can experiment with *Graphics[]* to create a completely customized graphic.

Basic phylogenetic functions

CtoTree[*matrix, labels*]

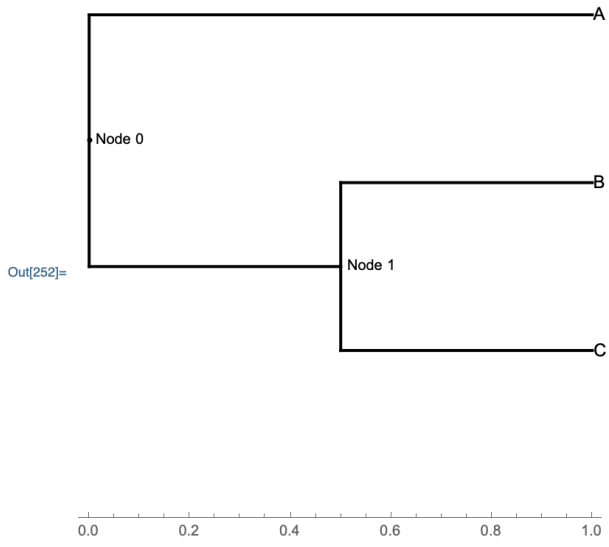
This function transforms a phylogenetic covariance matrix (known as the C matrix by many authors (*e.g.*, Revell and Harmon, 2008) and as the *VarY* by Martins and Hansen (1997) into a Newick format tree.

Arguments:

- *matrix* is a phylogenetic covariance matrix in which the diagonal elements are the distances between root and the respective tip and the off diagonal elements are the shared branch lengths of two tips. This format is produced by the *PhylogeneticMatrices[]* function (element 1) of this package, minus the labels
- *labels* is a vector of taxon names in the same order as the matrix.

Example:

```
In[249]:= Cmat // MatrixForm
Out[249]//MatrixForm=
          1   0    0
          0   1.   0.5
          0   0.5  1.

In[251]:= newtree = CtoTree[Cmat, {"A", "B", "C"}]
Out[251]= (A:1,(B:0.5,C:0.5):0.5);

In[252]:= DrawNewickTree[newtree]
```



Out[252]=

DrawNewickTree[*tree* (, *AspectRatio*)]

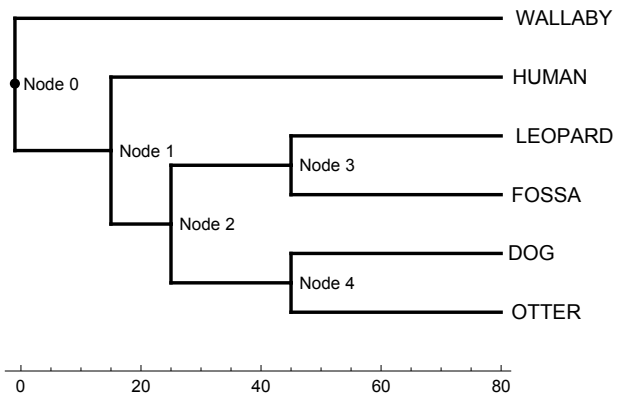
Parses a Newick tree string and renders it graphically as a tree with the tips and nodes labeled and the branches drawn proportional to the branch lengths in the file. This function requires branchlengths.

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *AspectRatio* is an optional real positive number that specifies the aspect ratio of the graphical output of the function

Example:

```
DrawNewickTree[tree]
```



GetFullLineage[*taxon, treetable*]

This function traces the branches of a tip taxon to the base of the tree from a tree table. It returns the branch entries of the table. This function is used by other functions in the package, but it may be useful on its own.

Arguments:

- *taxon* is the name of a taxon as a string
- *treetable* is is a tree that has been converted to a four-columntable using the TreeToTable[] function.

Example:

```
GetFullLineage[“Mustela_nigripes”,TreeToTable[MyTree]]

{{"Mustela_nigripes", "Node 39", 2.00462, 1}, {"Node 39",
"Node 38", 1, 0}, {"Node 38", "Node 36", 3.98236, 0},
{"Node 36", "Node 35", 0.659135, 0}, {"Node 35", "Node 33",
```

```
1.10872, 0}, {"Node 33", "Node 30", 3.06108, 0}, {"Node
30", "Node 29", 0.516018, 0}, {"Node 29", "Node 28",
5.23512, 0}, {"Node 28", "Node 17", 10.6648, 0}, {"Node
17", "Node 15", 7.72958, 0}, {"Node 15", "Node 7", 1.29123,
0}, {"Node 7", "Node 0", 4.74731, 0}}
```

IndependentContrasts[*tree, trait*]

This function calculates standardized independent contrasts (Felsenstein, 1985, 2004) for a single continuous trait on a tree.

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by *TreeToTable*[].
- *trait* is a two column matrix with taxon names in the first column and trait values in the second column. Taxon names must match the names in *tree*.

Example:

```
contrasts = IndependentContrasts[tree, trait];
```

Kappa[*tree, trait*]

This function estimates Blomberg's *K* (Kappa) for a trait given a tree (Blomberg *et al.*, 2003). *K* takes on a value of 1 when the tree explains as much trait variance as expected under Brownian motion and it approaches 0 when the tree explains none of it. Note that *K* can be larger than 1 if trait covariances have greater fidelity with the tree than expected under Brownian motion, which can happen under directional or diversifying models of evolution.

K can be interpreted as the proportion of trait variance that is explained simply by phylogenetic structure under a Brownian motion model of evolution because it is the ratio of observed MSE_0/MSE to the MSE_0/MSE, where MSE_0 is the mean squared error of the raw trait data around the value of the base node and *MSE* is the mean squared error of the trait data using the phylogenetic covariance matrix of the tree (Blomberg *et al.*, 2003).

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by *TreeToTable*[].
- *trait* is a two column matrix with taxon names in the first column and trait values in the second column. Taxon names must match the names in *tree*.

Example:

```
Kappa[MyTree, MyTrait]

0.890475
```

KappaMultivariate[*tree, trait*]

This function estimates Adams' (2014) multivariate *K* (Kappa) for a set of traits given a tree. Adams' multivariate *K* is analogous to Blomberg's *K* (Blomberg *et al.*, 2003) in that *K* takes on a value of 1 when the tree explains as much trait variance as expected under Brownian motion and it approaches 0 when the tree explains none of it. Note that *K* can be larger than 1 if trait covariances have greater fidelity with the tree than expected under Brownian motion, which can happen under directional or diversifying models of evolution.

Adams' multivariate *K* is designed for high-dimensional multivariate datasets, particularly those used in geometric morphometric analysis (Adams, 2014).

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by *TreeToTable*[].
- *trait* is a two column matrix with taxon names in the first column and trait values in the second column. Taxon names must match the names in *tree*.

Example:

```
KappaMultivariate[MyTree, MyTrait]

0.900367
```

LambdaMultiplication[*C, λ*]

This function multiplies a tree matrix by a scaling factor Pagel's λ, which usually has a value between 0 and 1, but which can take on values greater than 1 up to the maximum (tree height / highest node). It returns a new *C* matrix with adjusted branch lengths. Entering 0 will produce a star phylogeny, 1 will return the original matrix.

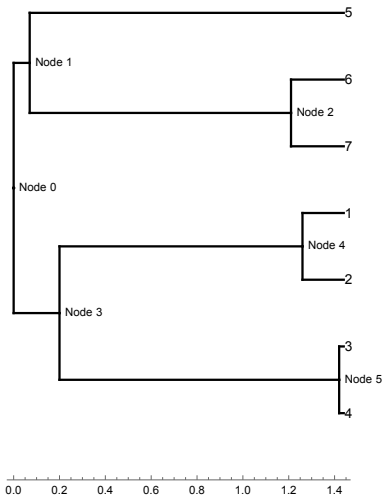
Arguments:

- *C* is a phylogenetic covariance matrix of the sort produced by *PhylogeneticMatrices*[] of the VarY variety stripped of its labels.
- *λ* is a number between 0 and 1 (or up to the maximum as described above.

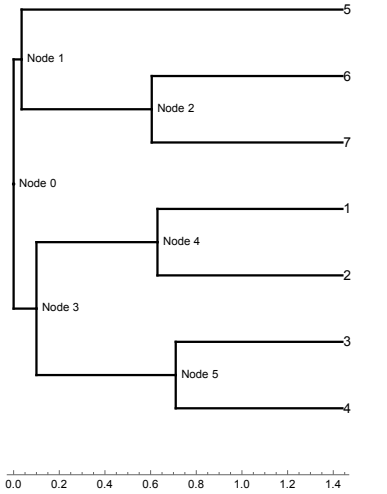
Example:

```
newC = LambdaMultiplication[C, 0.5];
```

```
DrawNewickTree[CtoTree[C,tiplabels]]
```




```
DrawNewickTree[CtoTree[newC,tiplabels]]
```



LineageEvolution[*n, "r"->1, "d" -> 0, "s" -> 0, "a" -> 0, "t" -> 1*]

This function can be used to model trait evolution as Brownian motion, directional, or Ornstein-Uhlenbeck (OU) process.

The default options model evolution as a Brownian motion random walk for *n* generations with a step rate of "*r*". The walk performed by this function has a stochastically variable step rate, with the change at each generation is drawn from a normal distribution with mean of zero and standard deviation of "*r*". By default this value is 1.

Directionality is added to the process by setting "*d",* which adjusts the mean of the step rate. A positive "*d"* will produce directional evolution in the positive direction, a negative

"*d"* will do the opposite. By default, "*d*" is 0. Note that *d* and *r* combine to produce the step rate.

Stabilizing processes can be added by setting the "*a"* and "*t"* options. "*t"* is the target trait value, which can be thought of as the center of an adaptive peak. Selection moves the trait toward this mean. "*a"* is a value between 0 and 1 that represents the strength of the selection, also equivalent to the narrowness of the adaptive peak. If "*a"* is 0 then there is no effect and the process behaves like ordinary Brownian motion or directional evolution (the default setting is 0). If "*a"* is 1, then the OU process has full strength.

Note: this function returns results as a list of pairs of numbers, the first of which is the step number and the second is the trait value.

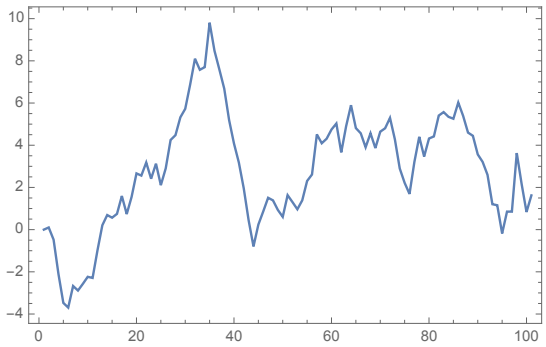
Arguments:

- *n* is the number of steps (generations) in the random walk.
- *"r"* is a positive real number for the per-step rate of change.
- *"d"* is a real number for the mean of the step variance (rate). By default this value is zero. Values other than zero add directionality to the evolutionary pattern.
- *"s"* is a real number that specifies the starting value of the trait. By default it is set to 0.
- *"a"* is a real number that defines the strength of the constraining force in an OU process. By default it is set to 0.0, which means that OU has no effect. A value of 1.0 gives the OU process its full strength.
- *"t"* is a real number that defines the target trait value in an OU process, which can be thought of as the center of the adaptive peak. The constraining process of OU pushes the phenotype toward this peak.

Examples:

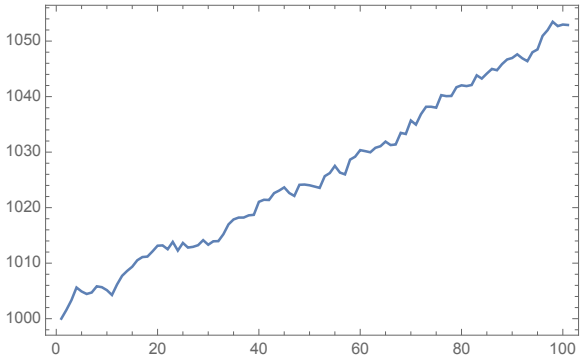
Random Walk:

```
lineage = LineageEvolution[100];
ListPlot[lineage, Axes->False, Frame->True, Joined->True,
PlotRange->All]
```



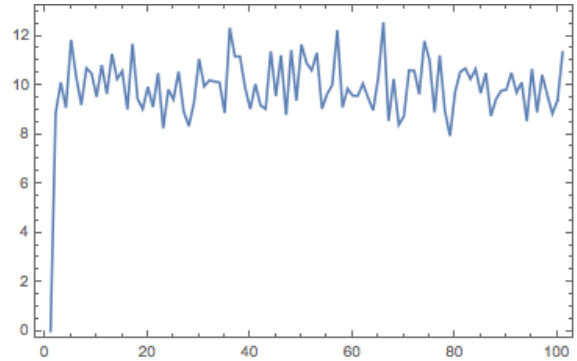
Directional Selection:

```
lineage = LineageEvolution[100, "r"->1, "m"->0.5, "s"-
>1000];
ListPlot[lineage, Axes->False, Frame->True, Joined->True,
PlotRange->All]
```



OU Process:

```
lineage = LineageEvolution[100, "r"->1, "a"-> 1, "t"->10];
ListPlot[lineage, Axes->False, Frame->True, Joined->True,
PlotRange->All]
```



MakePHYLIPLabels[*labels*]

Reformats a list of labels to be compatible with PHYLIP. Labels that are longer than 10 characters are truncated, and those that are shorter than 10 characters are padded with spaces.

Arguments:

- *labels* is a list of strings to be used as OTU labels for PHYLIP.

Example:

```
phyliplabels = MakePHYLIPLabels[labels]

{"WALLABY   ", "LEOPARD   ", "HUMAN     ", "OTTER     ",
"FOSSA     ", "DOG       "}
```

MapTraitsOntoTree[*tree, trait (, "ColorPattern", "TraitRange", "NodeLabels")*]

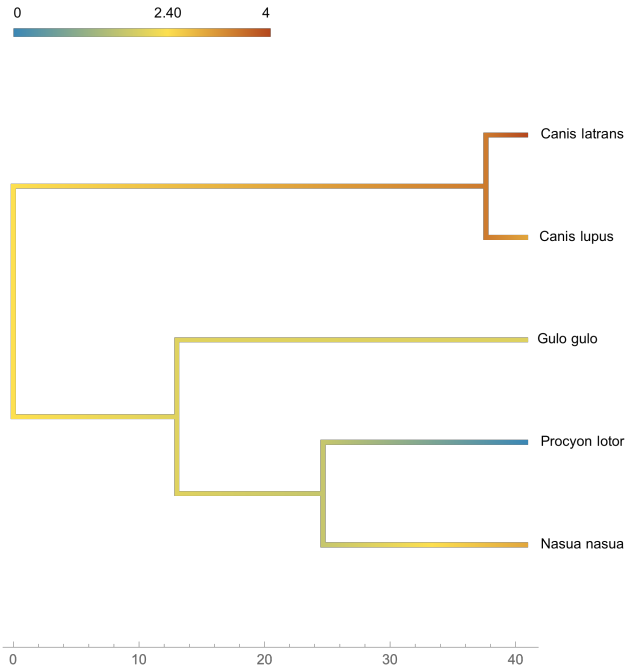
This function maps trait values onto a phylogenetic tree assuming a Brownian motion model of evolution. The output is a tree with branches color-coded to correspond to the estimated values of the trait at its nodes and along its branches. By default, the values are colored with a blue to yellow to orange spectrum with pure yellow representing the ancestral value, pure blue the numerically smallest trait value in the dataset, and pure red the largest value. Different color schemes may be selected by setting the optional "ColorPattern" argument with a list of three colors that will be substituted for blue, yellow, and red. And by default the color scheme is scaled between the maximum and minimum trait values in the data set. User may force a different scaling using the "TraitRange" option.

Arguments:

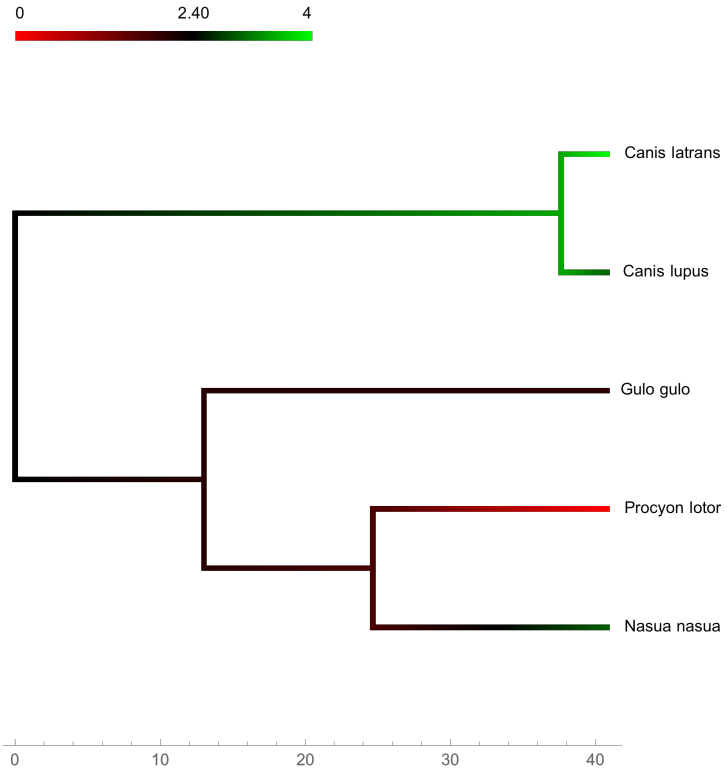
- *tree* is a Newick-format tree in the format of ReadNewick[]
- *trait* is a matrix with a vector of taxon names in the first column and trait values in the second column (note that the taxon names must be identical to those in the tree)
- *"ColorPattern"* is an optional list of three *Mathematica* colors (e.g., "ColorPattern" -> {Green, Yellow, RGBColor[0.995666, 0.760983, 0.611231]})
- *"TraitRange"* is an optional list of the minimum and maximum values for the color scaling (e.g. "TraitRange" -> {0, 1}).
- *"NodeLabels"*, when set to True, prints labels for nodes.

Examples:

```
MapTraitsOntoTree[MyTree, MyData]
```




```
MapTraitsOntoTree[MyTree, MyData, "ColorPattern"->
{Red,Black,Green}]
```



MeanTreeDistance[*taxa, tree*]

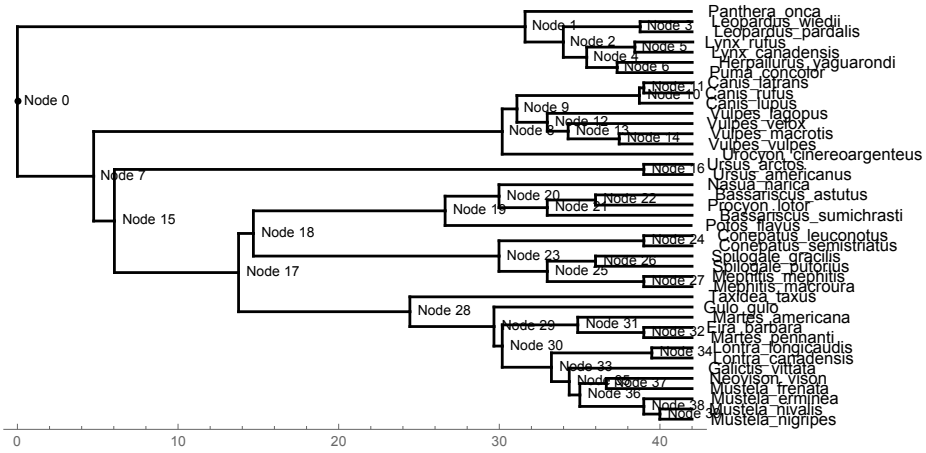
This function calculates mean tree distance for a subset of taxa in a larger clade. The metric is the mean pairwise distance (in branch length units) between the selected of taxa (Polly *et al.*, 2017) and is analogous to Webb's (2000) mean pairwise nodal distance. This function assumes that the input tree is ultrametric because it uses depth of shared ancestry of first taxon in the matrix as measure of depth of the tree as a whole.

Arguments:

- *taxa* is a list of taxon names as strings (all taxa must be in the tree and none must be entered more than once)
- *tree* is a phylogenetic tree in Newick format.

Example:

```
DrawNewickTree[CarnivoreTree]
```




```
MeanTreeDistance[{"Panthera_onca","Lynx_rufus","Vulpes_velo
x"}, CarnivoreTree]

{3, 31.4208}
```

OmearaAncestor[*tree, trait*]

This function estimates the ancestor value(s) for the base node in a tree using a Brownian motion model. This function is based on the least squares algorithm described by O'Meara *et al.* (2006), Revell and Harmon (2008) and others. The function returns an estimate of the ancestral value of each trait for the base of the tree.

Arguments:

- *tree* is a Newick-format tree in the format of ReadNewick[]

- *trait* is a matrix with a vector of taxon names in the first column and trait values in the second and subsequent columns (note that the taxon names must be identical to those in the tree)

Example:

```
MyAncestor = OmearaAncestor[MyTree, MyTrait]

{ -4.91801, -1.59127}
```

OmearaRates[*tree, trait*]

This function estimates a rate (or rate matrix) for one more traits using a Brownian motion model. The function is based on the algorithms described by O'Meara *et al.* (2006), Revell and Harmon (2008) and others. For multiple traits, the rate matrix diagonal contains the σ^2 rates for each trait and the off-diagonal elements are rate covariances, which are essentially "directionality" terms with respect to a bivariate or multivariate trait space.

Arguments:

- *tree* is a Newick-format tree in the format of ReadNewick[]
- *trait* is a matrix with a vector of taxon names in the first column and trait values in the second and subsequent columns (note that the taxon names must be identical to those in the tree)

Example:

```
OmearaRates[MyTree, MyTrait]

{{0.880145, -0.113647}, {-0.113647, 1.13332}}
```

PagelsLambda[*tree*, *trait*]

This function estimates Pagel's *λ* (lambda) (Pagel, 1993) using the likelihood method described by Freckleton et al. (2002) and partially following Liam Revell's *phylosig()* code in his package *phylotools* (Revell, 2011). It uses two functions, one that calculates the probability of the traits given a phylogenetic covariance and lambda, and another that maximizes this function for the traits.

Pagel's *λ* is a scaling factor that transforms the branches of a tree from their native lengths to a perfect star phylogeny. Normally *λ* ranges between 0 (no phylogenetic structure to the trait value) to 1 (phylogenetic structure in the trait that is consistent with Brownian motion), but can be higher than 1 if the phylogenetic covariance in the trait is stronger than expected under Brownian motion. Note that *λ* should not be interpreted as a proportion of variance explained by phylogeny, which is what Blomberg's *K* reports (see Polly et al., 2017).

The function returns estimated value of *λ*, an estimated ancestral value of the trait at the base of the tree, and an estimated squared rate parameter (σ^2), both of which are estimated after adjusting for the level of *λ*. Note that likelihood maximization in this function can take a while to execute for large trees.

Arguments:

- *Tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *Trait* is a 2 x *n* matrix with tip labels in the first column and values of a continuous trait in the second column. Tip labels must match the labels used in the tree.

Example:

```
lamd = PagelsLambda[tree, trait]
```

```
= {λ → 1.01, LogLikelihood → -9.932577322, Ancestral Value → 1.82913, σ2 → 0.38585}
```

PhylogeneticMatrices[*tree*]

Creates the phylogenetic matrices needed for ancestral node reconstruction and other comparative statistical methods based on Generalized Linear Models (GLM). The function produces three matrices: the first describes the shared ancestry of the tip taxa (VarY), the second describes the shared ancestry between the tips and the nodes (VarAY), and the third describes the shared ancestry of the nodes (VarA). Matrices are returned with row and column labels. These matrices are described by Martins and Hansen (1997).

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].

Example:

```
{varY, varAY, varA} = PhylogeneticMatrices[tree];
```

PhylogeneticRegression[*traits, tree*]

This function performs a phylogenetic regression using the algorithm of Martins and Hansen (1997). It returns the slope an intercept of a linear regression model adjusted for the phylogenetic covariance of the taxa, R^2 for the phylogenetically adjusted relationship between the two variables, an ANOVA table with p-value, and the predicted values for trait Y for each taxon.

Arguments:

- *traits* is a matrix with three columns, the first with taxon names as strings, the second with trait values for the independent (X) variable, and the third with trait values for the dependent (Y) variable. Note that the taxon names must precisely match the names in the tree.
- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].

Example:

```
PhylogeneticRegression[MartinsTraits, MartinsTree]
```

```
Out[86]= {{Intercept → 1.28996, Slope → 0.0162722}, R-Squared → 0.14,

                              DF    SS          MS          F         P
          ANOVA Table → Model 1     0.922548    0.831065    1.1101    0.32707
                        Error 7     5.81745     0.8425                      ,
                        Total 8     6.74

         SE Regression → 0.911627, Predicted Values →

          {2.59174, 2.91719, 2.02222, 2.72192, 2.15239, 2.2663, 2.83583, 2.47784, 2.78701}}
```

PhylogeneticRegressionWithLambda[*traits, tree, λ*]

This function performs a phylogenetic regression using the algorithm of Martins and Hansen (1997) and is identical to *PhylogeneticRegression[]* except that the user may specify a Pagel's λ (lambda) value that alters the lengths of the internal branches to conform with a non-Brownian motion model of evolution. The function returns the slope an intercept of a linear regression model adjusted for the phylogenetic covariance of the taxa, R^2 for the phylogenetically adjusted relationship between the two variables, an ANOVA table with p-value, and the predicted values for trait Y for each taxon.

Arguments:

- *traits* is a matrix with three columns, the first with taxon names as strings, the second with trait values for the independent (X) variable, and the third with trait values for the dependent (Y) variable. Note that the taxon names must precisely match the names in the tree.
- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *λ* is a value of Pagel's λ that is usually between 0 and 1 and which scales the internal branch lengths accordingly (see Martins and Hansen, 1997).

PhylogeneticsVersion[]

Prints the version number and citation for the current installation.

Example:

```
PhylogeneticsVersion[]

Phylogenetics for Mathematica 6.8
(c) P. David Polly, 7 June 2023
```

PruneTree[*taxa, tree*]

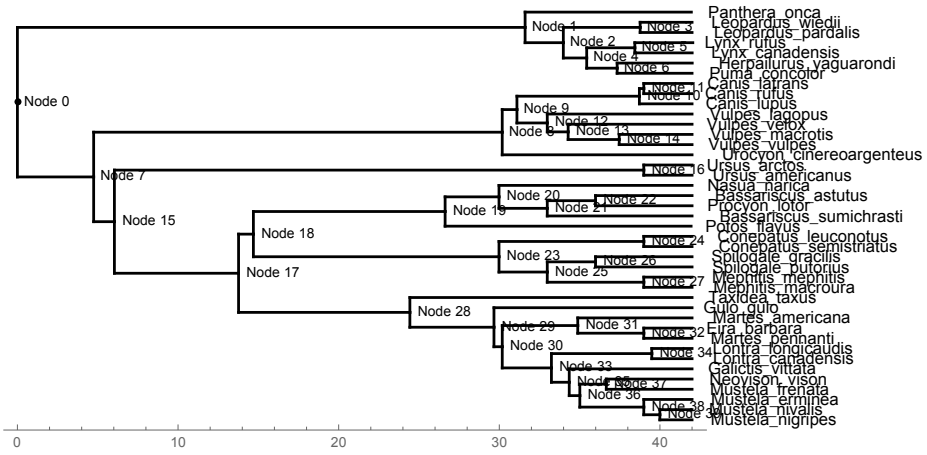
This function takes returns a Newick tree composed of a subset of taxa selected from a larger tree. The original branch scaling (presumed to be in absolute time units) is retained in the pruned tree.

Arguments:

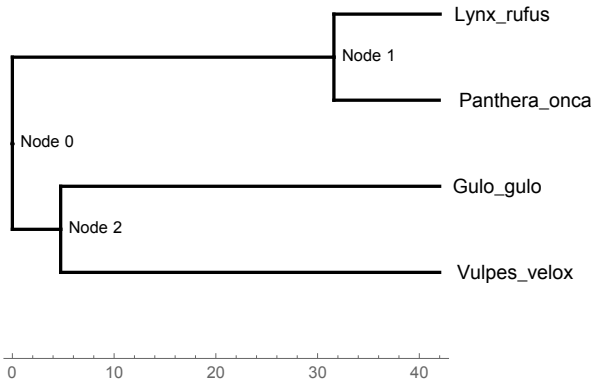
- *taxa* is a list of taxon names as strings (all taxa must be in the tree and none must be entered more than once)
- *tree* is a phylogenetic tree in Newick format.

Example:

```
DrawNewickTree[CarnivoreTree]
```




```
myclade =
{"Panthera_onca","Lynx_rufus","Vulpes_velox","Gulo_gulo"};
smallertree = PruneTree[myclade, CarnivoreTree]

((Lynx_rufus:10.4404,Panthera_onca:10.35):31.6062,(Gulo_gul
o:37.2527,Vulpes_velox:37.2527):4.74731);

DrawNewickTree[smallertree]
```



RandomWalk[*n, i*]

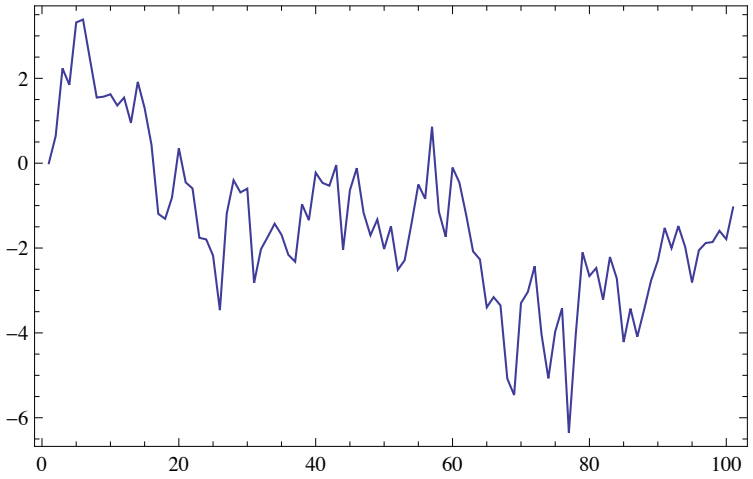
By default, this function performs a Brownian motion random walk for *n* generations with a step rate of *i*. The walk performed by this function has a stochastically variable rate, with the change at each generation is drawn from a normal distribution whose mean is zero and whose standard deviation equals *i*.

Arguments:

- *n* is the number of steps (generations) in the random walk.
- *i* is a positive real number for the per-step rate of change.

Example:

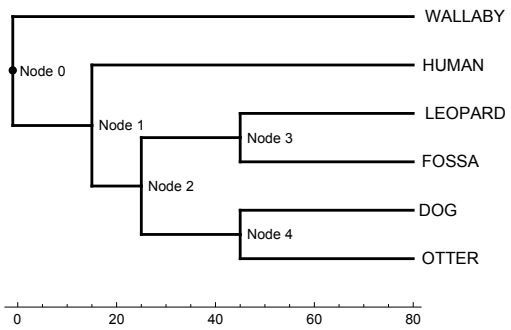
```
walk = RandomWalk[100, 1];

ListPlot[walk, Axes->False, Frame->True, Joined->True,
PlotRange->All]
```



ReadNewick[*filename*]

Reads one or more Newick format trees from a text file and prepares them for use by removing carriage returns, line feeds, and other extraneous characters. A Newick tree is represented as a string consisting of tip names, branch lengths (separated from tip names or nodes by colons), and parentheses. Newick tree formats are described in detail in Felsenstein (2004). For example, this Newick string describes the following tree:

```
(WALLABY:80,(HUMAN:65,((LEOPARD:35,FOSSA:35):20,(DOG:35,OTT
ER:35):20):10):15);
```



Arguments:

- *filename* is the name, with path if needed, of a text file containing only Newick format trees.

Example:

```
tree=ReadNewick[“/Users/Data/mytree.tre”]

(WALLABY:80,(HUMAN:65,((LEOPARD:35,FOSSA:35):20,(DOG:35,OTT
ER:35):20):10):15);
```

ReadPhylip[*filename*]

This function reads in a Phylip formatted cladistic character matrix and returns it as an array with the taxon name in the first column and the characters separated into columns.

Arguments:

- *filename* is the name, with path if needed, of a text file containing only Newick format trees.

Example:

```
data=ReadPhylip[“/Users/Data/infile.txt”]
```

ReconstructNodes[*tree, trait*]

This function returns the estimated rate of evolution of the trait on the tree, a list of reconstructed values of trait Y for each node on the tree, and the standard deviation of the node estimate due to variation in the evolutionary process.

The per-step rate of evolution is estimated as the square-root of the mean of the squared standardized independent contrasts (Felsenstein, 1985), which is equivalent to the method described by Martins (1994) when a Brownian motion model of evolution is assumed. Note that the rate parameter returned here is the per-step rate per unit of branch length in the tree, not the squared rate (the latter also referred to as the step variance or σ^2). The rate is returned in the first matrix of the results.

The ancestral node values are reconstructed using the generalized linear model (also known as phylogenetic generalized least squares, or PGLS) described by Martins and Hansen (1997; see also Rohlf, 2001). This method assumes a Brownian motion model of evolution and provides the same estimates as squared-change parsimony (Maddison, 1991) and maximum-likelihood (Schluter *et al.*, 1997). The node values are returned in the second matrix of the results.

The standard deviation of each node describes the uncertainty due to the evolutionary process, which in this case is assumed to be Brownian motion. The node values are distributed normally (Felsenstein 1973, 2004). The standard deviations reported by this function are the square roots of the node variances based on the corrected equations given by Rohlf (2001) and calculated by successive re-rooting of the tree (Garland and Ives, 2000). The standard deviations of the nodes are returned in the third matrix of the results.

Note that the rate, the node reconstructions, and the standard deviations of the nodes are all parameters which have uncertainties due to estimation. The uncertainty is a complicated function of the rate, the number of branches on the tree, the topology and lengths of the branches, and the mode of evolution. One can explore the magnitude of the estimation errors by Monte Carlo simulation of data on the tree using the *SimulateContinuousTraitOnTree[]* function that is included in this package.

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *trait* is a 2 x *n* matrix with tip labels in the first column and values of a continuous trait in the second column. Tip labels must match the labels used in the tree.

Example:

```
{rate, nodes, sd} = ReconstructNodes[MyTree, MyTrait];
```

ReconstructNodesSimple[*tree, trait*]

This function is the same as *ReconstructNodes[]* in that it returns the reconstructed values of a trait for each node on a tree, but it uses a more efficient algorithm and it does not return the rate of evolution or standard errors.

Arguments:

- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *trait* is a 2 x *n* matrix with tip labels in the first column and values of a continuous trait in the second column. Tip labels must match the labels used in the tree.

Example:

```
ReconstructNodes[MyTree, MyTrait]
```

RerootTree[*tree, node*]

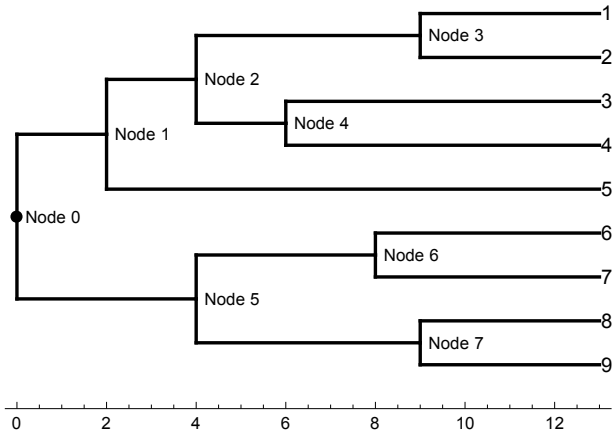
This function reroots a Newick tree at a specified node.

Arguments:

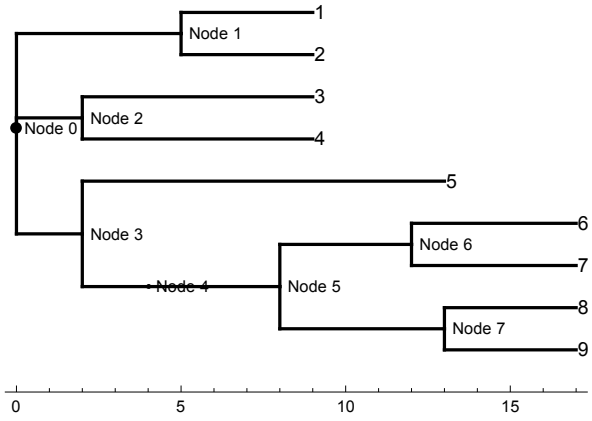
- *tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *node* is the name of the node where the new tree will be rooted. The available names can be displayed by plotting the tree with DrawNewickTree[tree].

Example:

```
DrawNewickTree[tree]
```



```
newtree = RerootTree[tree, "Node 2"];
DrawNewickTree[newtree]
```



SharedPhylogeneticHistory[*taxa, tree*]

This function measures several aspects of phylogenetic community structure from a list of taxa and a tree. The metrics reported by the function are as follows:

1. the name of the last common ancestor of the taxa in the selected set;
2. the number of taxa in the selected set as a proportion of the total in the clade,
3. the age of their last common ancestor proportional to the total time elapsed between base of tree and latest occurring tip; and
4. the lineage length shared by the taxon set as a proportion of the lineage length of the entire tree.

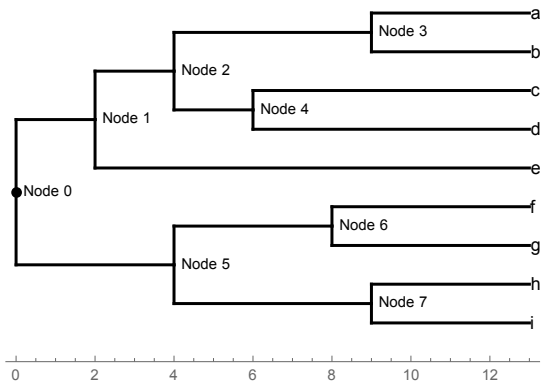
These metrics are useful for studying phylogenetic community structure (e.g., Webb, 2000; Kraft and Ackerly, 2010), but are not identical to the metrics used most frequently in the literature (in large part because this package is designed from a paleontological perspective and it is assumed that trees are calibrated in absolute time, e.g., millions of years). Some of the metrics returned by this function were combined with permutation tests to identify areas of North America where carnivoran communities are more closely related than expected by chance (Polly *et al.*, 2017).

Arguments:

- *taxa* is a list of taxon names as strings (all taxa must be in the tree and none must be entered more than once)
- *tree* is a phylogenetic tree in Newick format.

Example:

```
DrawNewickTree[MartinsTree]
```




```
SharedPhylogeneticHistory[{"b", "c", "e"}, MartinsTree]

{Node 1, 0.33, 0.15, 0.41}
```

SimulateContinuousTraitsOnTree[*Tree, Rates (,StartValue, IncludeNodes)*]

This function simulates the evolution of a continuous trait (or traits) on a tree using a Brownian motion model of evolution. By default the trait starts with a value of 0.0 at the root of the tree (this can be changed by specifying a different value as the third input parameter of the function). Evolution is simulated along each branch by Brownian motion using the rate specified by the second input parameter. Note that *Rate* is the per-step rate of change, not the squared per-step rate of change (σ^2 of authors). The function returns a list of node and tip labels followed by the simulated trait value.

Arguments:

- *Tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *Rate* is the amount of trait change per unit of branch length (as specified in the tree). The *Rate* parameter should be a single number if only one trait is being simulated or a list of numbers enclosed in curly brackets if multiple traits are being simulated.
- *StartValue* is the optional starting value of the trait at the root of the tree (by default the value is 0). If more than one trait is being simulated, then StartValue will be a list of starting values enclosed in curly brackets equal in number to the list of rates.
- *IncludeNodes* is the optional parameter that indicates whether simulated node values should be returned (1) or not returned (0, default).

Example 1: Simulate one trait with step rate of 1.

```
SimulateContinuousTraitsOnTree[myTree, 1]
```

Example 2: Simulate three traits with rates of 1

```
SimulateContinuousTraitsOnTree[myTree, {1,1,1}]
```

Example 3: Simulate two traits with ten steps per branch length unit:

```
SimulateContinuousTraitsOnTree[myTree, {1,1},10]
```

Example 4: Simulate one trait with 10 steps per branch length unit with starting value of 0 and reporting node values:

```
SimulateContinuousTraitsOnTree[myTree, 1, 10, 0, 1]
```

SimulateCorrelatedTraitsOnTree[*Tree, CovarianceMatrix (, StartVector, IncludeNodes)*]

This function simulates the evolution of *k* continuous correlated traits on a tree using a Brownian motion model of evolution. By default each trait starts with a value of 0.0 at the root of the tree (this can be changed by specifying a different value as part of a vector in the third input parameter of the function). The number of traits is determined by the covariance matrix, as is the rate. The step variance rate for each trait (σ^2 of authors) is the variance recorded for that trait in the diagonal of the covariance matrix. The function returns a list of node and tip labels followed by vectors containing the simulated trait values.

Arguments:

- *Tree* is a string containing a tree with branch lengths in Newick format or a tree table of the sort produced by TreeToTable[].
- *CovarianceMatrix* is a square symmetric matrix describing the variance and covariances of the trait. Note that the variances are used as the rate parameters in the simulation.
- *StartVector* is an optional vector containing the starting value of each trait at the root of the tree (by default the value is 0)
- *IncludeNodes* is the optional parameter that indicates whether simulated node values should be returned (1) or not returned (0, default)

Example:

```
SimulateCorrelatedTraitsOnTree[myTree, {{.9, .3},{.3, .5}}]
```

TableToTree[*table*]

Parses a tree table and returns a Newick format tree in a single string with no spaces or line feeds. The table should be in the same format as the ones returned by the *TreeToTable[]* function.

Arguments:

- *table* has one row for each branch in the tree and four columns: the first column contains the name of the descendant tip or node, the second column contains the name of the ancestor tip or node, the third column contains the length of the branch, and the fourth column indicates whether the branch ends in a tip (1) or a node (2).

Example:

```
myTable = {{"Descendant", "Ancestor", "Branch Length",
"Tip?"}, {"1", "Node 0", 1,  1}, {"2", "Node 1", 1, 1},
{"3", "Node 1", 1, 1}, {"Node 1", "Node 0", 1,  0}}

TableToTree[myTable]

(1:1,(2:1,3:1):1);
```

ThreeModelTest[*lineage*]

This function fits three evolutionary models to data derived from an evolutionary lineage following Hunt (2006). Using Hunt's terminology, it fits an unbiased random walk (URW) that is purely Brownian motion with no directional component, a general random walk (GRW) that has a directional component, and a stasis model (Stasis) to a set of trait data from a lineage sampled at different points along a lineage.

Arguments:

- *lineage* is a data matrix with time in the first column, trait values in the second column, sample error as a variance in the third column, and number of individuals in the sample in the fourth column. If only the first two columns are given then the algorithm assumes there is no sampling error. Note that time should be given in absolute units of the stratigraphic age of each interval, either in ascending or descending order. The function *LineageEvolution[]* returns data in this format.

Example:

```
ThreeModelTest[lineage]
```

Best model	URW
N	39
$\hat{\mu}_{step}$	0.166
$\hat{\sigma}^2{}_{step}$	0.034
$\hat{\theta}$	1.384
$\hat{\omega}$	0.004
AIC_c URW	−124.270
AIC_c GRW	−122.860
AIC_c Stasis	−96.080
wt URW	0.670
wt GRW	0.330
wt Stasis	5.064×10^{-7}

TreeToTable[*tree*]

Parses a Newick format tree, supplied as a single string with no spaces or line feeds (such as the string given by the function ReadNewick[]) and returns a table with one row for each branch in the tree and four columns: the first column contains the name of the descendant tip or node, the second column contains the name of the ancestor tip or node, the third column contains the length of the branch, and the fourth column indicates whether the branch ends in a tip (1) or a node (2). This table (minus its header row) is used by other functions in this package, notably PhylogeneticMatrices[].

Arguments:

- *tree* is a Newick format tree entered as a single string, similar to the format produced by ReadNewick[].

Example:

```
myTree = "(1:1,(2:1,3:1):1);"

TreeToTable[myTree]  //MatrixForm
```

Descendant	Ancestor	Branch Length	Tip?
1	Node 0	1	1
2	Node 1	1	1
3	Node 1	1	1
Node 1	Node 0	1	0

UltrametricizeTree[*tree*]

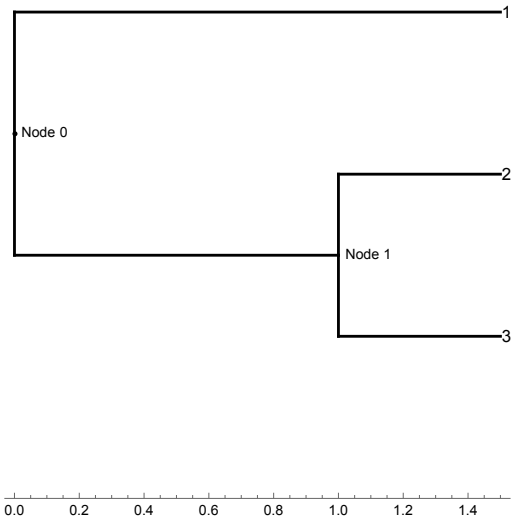
This function takes as input a non-ultrametric tree and creates an ultrametric tree by lengthening all tip branches to be coeval with the latest occurring tip.

Arguments:

- *tree* is a Newick format tree entered as a single string, similar to the format produced by ReadNewick[] or a tree table in the format output by TreeToTable[] (minus the header row).

Example:

```
myTree = "(1:1,(2:0.1,3:0.5):1);"

myUltrametricTree = UltrametricizeTree[myTree];

DrawNewickTree[myUltrametricTree]
```



Acknowledgements

Jim Rohlf, Emilia Martins, and Gene Hunt helped interpret some of their equations. Joe Felsenstein gave advice on converting branch lengths to variance units for some of the earliest versions of functions contained in this package. Liam Revell discussed the behavior of Blomberg's Kappa and Pagel's Lambda. Michelle Lawing, Aida Gómez Robles, Jesualdo Fuentes Gonzales, and members of my G562 course on geometric morphometrics have made suggestions that have improved the accuracy and operation of these functions.

Funding for development for this package has been provided by grants NSF EAR 1338298, the Robert R. Shrock fund at Indiana University, the Yale Institute for Biospheric Studies, and the Lilly Endowment through its support for the Indiana University Pervasive Technology Institute and the Indiana METACyt Initiative.

Bibliography

Blomberg, S. P., Garland, T., and Ives, A. R. 2003. Testing for phylogenetic signal in comparative data: behavioral traits are more labile. *Evolution*, **57:** 717-745.

Felsenstein, J. 1973. Maximum-likelihood estimation of evolutionary trees from continuous characters. *American Journal of Human Genetics*, 25: 471-492.

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

Felsenstein, J. 2004. *Inferring Phylogenies*. Sinauer Associates, Sunderland, Mass.

Freckleton, R. P., Harvey, P. H., and Pagel, M. 2002. Phylogenetic analysis and comparative data: a test and review of evidence. *Am. Nat.*, **160:** 712-726.

Garland, T. Jr. and A. R. Ives. 2000. Using the past to predict the present: confidence intervals for regression equations in phylogenetic comparative methods. *American Naturalist*, 155: 346-364.

Hunt, G. 2006. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology*, 32: 578-601.

Maddison, W. P. 1991. Squared-change parsimony reconstructions of ancestral states for continuous-valued characters on a phylogenetic tree. *Systematic Zoology*, 40: 304—314.

Martins, E. P. 1994. Estimating the rate of phenotypic evolution from comparative data. *American Naturalist*, 144: 193-209.

Martins, E. P. and T. F. Hansen. 1997. Phylogenies and the comparative method: a general approach to incorporating phylogenetic information into the analysis of interspecific data. *American Naturalist*, 149: 646-667.

O'Meara, B C., C. Ané, M. J. Sanderson, and P. C. Wainwright. 2006. Testing for different rates of continuous trait evolution using likelihood. *Evolution*, **60:** 922-933.

Pagel, M. 1993. Inferring the historical patterns of biological evolution. *Nature*, **401:** 877-884.

Polly, P. D., J. Fuentes-Gonzales, A. M. Lawing, A. K. Bormet, and R. G. Dundas. 2017. Clade sorting has a greater effect than local adaptation on ecometric patterns in Carnivora. *Evolutionary Ecology Research*, **18:** 61-95.

Revall, L. J. 2011. phytools: an R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, 3(2): 217-223.

Revell, L. J. and L. J. Harmon. 2008. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evolutionary Ecology Research*, 10: 311-331.

Rohlf, F. J. 2001. Comparative methods for the analysis of continuous variables: geometric interpretations. *Evolution*, 55: 2143-2160.

Schluter, D., T. Price. A. O. Mooers, and D. Ludwig. 1997. Likelihood of ancestor states in adaptive radiation. *Evolution*, 51: 1699-1711.

Webb, C. O. 2000. Exploring the phylogenetic structure of ecological communities: an example for rain forest trees. *American Naturalist*, **156:** 145-155.