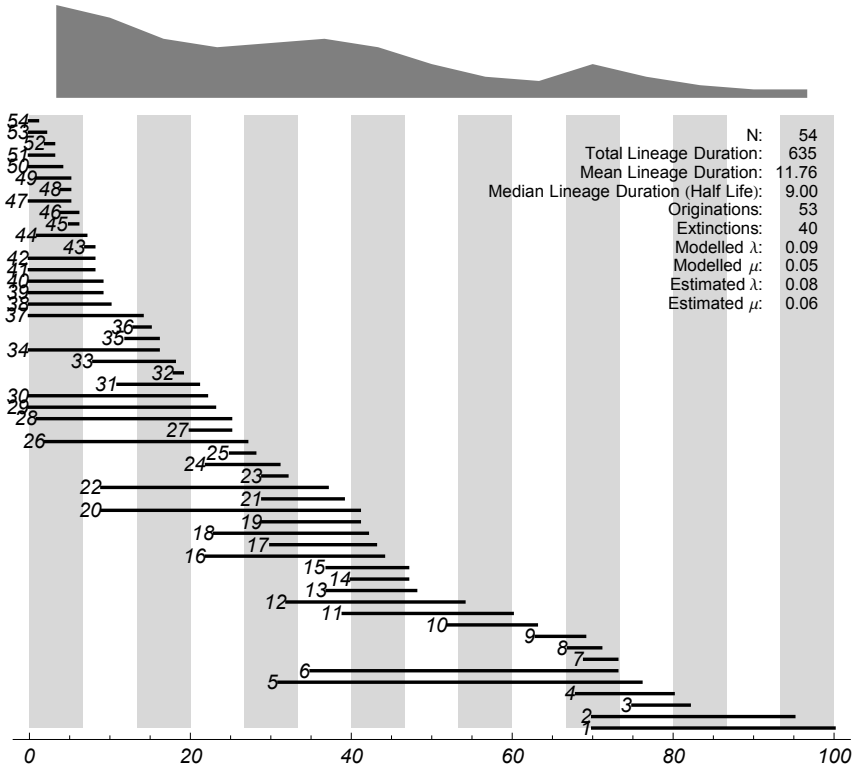
Quantitative Paleontology for Mathematica
User's Guide
Version 5.1

This package is used in Indiana University course
EAS-E 563 Quantitative Paleontology
(https://quantpaleo.earth.indiana.edu)



P. David Polly
Earth & Atmospheric Sciences
Indiana University
Bloomington, Indiana 47405 USA
https://pollylab.indiana.edu/
pdpolly@indiana.edu

Cite as: Polly, P.D. 2023. Quantitative Paleontology for Mathematica. Version 5.1.
https://github.com/pdpolly/Quantitative-Paleontology-for-Mathematica/

Table of Contents

Quantitative Paleontology for Mathematica User's Guide Version 5.1 1
New in 5.1 3
New in 5 3
New in 4 3
New in 3 3
Installing the Quantitative Paleontology Mathematica package 3
Using Mathematica 4
Retrieving data from MySQL in *Mathematica* 5
First connection 5
Working with established connections 5
Extracting data for Diversity functions 6
Functions 8
CohortAnalysis[data, censuspoint] 8
ConvertDIVAPoints[FileName] 9
ConvertParacladeToTree[myCreation] 10
CreationToTree (myCreation)] 10
Disparity[Data , (Ages, NumBins)] 11
DisparityWithStages[Data , Stages, StageOrders] 13
DiversityPlot[data, numbins (, λ, μ)] 14
DiversityTable[data, numbins] 16
ExportKML[FileName, Coordinates, Labels->labels, Descriptions->descriptions, ColorGroups->grouplabels, AgeBins->num bins, AgeData->ages] 16
GowerDistance[data] 19
LetThereBeLight[time, a, λ, μ (, maxprotect)] 19
LetThereBeLightWithPseudoExtinction [time, a, λ, μ (, maxprotect)] 20
LRI[*timeseries* (, sd)] 21
PrincipalCoordinatesAnalysis[*data (, "Gower")*] 22
QuantitativePaleontologyVersion[] 23
StigallAreaAnalysis[Data] 23
Acknowledgements 25
References 25

New in 5.1

CreationToTree[] Converts the output of *LetThereBeLightWithPseudoExtinction* to a Newick format tree (requires the *Phylogenetics for Mathematica* package)

New in 5

GowerDistance[] Produces a Gower distance matrix for data sets with variables that have incommensurate units or missing entries.

PrincipalCoordinatesAnalysis[] Performs a Principal Coordinates Analysis (PCO) on distance matrices using either Euclidean or Gower distances.

Disparity[] and DisparityWithStages[] Fixed several problems with the calculation of disparity statistics for time-binned data and introduced new distance measure.

New in 4

All functions return data using new methods that conserve memory.

Disparity[] Updates to the *Disparity* function calculate statistics over all dimensions of morphospace.

LetThereBeLight[] Function now stops when standing diversity reaches 10,000 lineages unless user specifies otherwise.

LetThereBeLightWithPseudoExtinction[] New function similar to LetThereBeLight, but with pseudo extinction of the parent lineage when speciation takes place.

ConvertParacladeToTree[] New function that converts the output of *LetThereBeLightWithPseudoExtinction[]* to Newick format so that it can be drawn or analysed using functions from the *Phylogenetics for Mathematica* package.

New in 3

LRI[] New function to estimate rate and mode of evolution using the Log-Rate/Log-Interval method of Gingerich (1993).

Installing the Quantitative Paleontology Mathematica package

1. Download the latest version of the package at https://github.com/pdpolly/Quantitative-Paleontology-for-Mathematica/ (right click on link to save as file)
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., “QuantPaleo”)
5. Once installed, enter the command “<<QuantPaleo`” to use the functions.

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. Mathematica 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.

Retrieving data from MySQL in *Mathematica*

Many of the functions in this package require large tables of data to be passed to them. One way to manage data is using MySQL, an open-source relational database from Oracle (http://www.mysql.com). Mathematica functions for interacting with SQL databases can be loaded from the DatabaseLink package:

```
<< DatabaseLink`
```

The database link functions allow you to interact with SQL-compliant databases on your own computer or on the network. Using the ODBC connections, you can even query Excel spreadsheets as though they were databases. Connections to MySQL are described here.

First connection

The first time you connect to MySQL you will need to define a new connection. The information about the connection can be stored using a name of your choice and used again any time you reconnect. Mathematica's database Connection Tool, which can be started by executing the *OpenSQLConnection[]* function. The Connection Tool will allow you to create a new connection, which you can give the name of your choice. You may optionally enter a description of the connection if you think you will forget what it does. The next frame asks if you want to create the connection only for you (User Level) or for anyone who uses your computer (System Level). Next you are asked to specify the type of database, which should be "MySQL(Connector/J)". After that you are asked to enter the server details, which are as follows if you are working with one of the AMP packages on your own computer:

Hostname:	localhost (or 129.0.0.1 if the localhost alias does not work on your machine)
Port:	3306
Username:	root
Password:	root (some AMP installations don't set a default password, in which case you should leave the password field blank)
Database:	can be left blank or you can choose from the list of databases in your MySQL

The next window allows you to set several advanced options, none of which need to be changed (though you can save the password by ticking the appropriate box, which saves you from having to enter it every time you make a query). Save this connection. After you have connected the first time, close the connection again by executing the *CloseSQLConnection[]* function.

Working with established connections

Once you have established a connection, you can perform most SQL operations from Mathematica. To do this you must open a connection and associate it with a variable of your choice. In the examples given here, the connection is stored in the variable "*conn*", but any variable name could be used. To open the connection, refer to it by name in the function *OpenSQLConnection[].* If you did not specify a database name, a username, or save a password, then you also specify those parameters in the function. The following example opens the "Felidae" database using the stored connection "mySQL":

conn = OpenSQLConnection["mySQL", Catalog -> "Felidae"];

The following example does the same thing, but specifies the username and password:

conn = OpenSQLConnection["mySQL", Catalog -> "Felidae",
Username -> "root", Password -> "root"];

You should close your connection when you are finished operating on the database using the *CloseSQLConnection[]* function and the variable where you stored the connection:

CloseSQLConnection[conn];

After opening the connection and before closing it you can execute other SQL functions. Querying is the most common kind of operation, but you can also create tables, delete data, replace data, or insert data. Mathematica has several SQL functions, the most generic of which is *SQLExecute[]* which allows a raw SQL command to be submitted directly to the database server. For example, the following function returns the entire contents of the table "occurrences" and stores them in the variable MyOccurrences:

*MyOccurrences = SQLExecute[conn, "SELECT * FROM occurrences;"];*

The first variable in the function is the open connection; the second is a string with the SQL statement enclosed in quotes. Note that the statement ends with a semicolon as part of the SQL syntax, and the entire line ends with a Mathematica semicolon that prevents the entire table from being printed to the screen. Any SQL statement can be executed this way, and Mathematica variables can be inserted into the statement with string joins. Note that if a numeric variable is inserted, it must first be converted to a string with the *ToString[]* function:

*MyOccurrences = SQLExecute[conn, "SELECT * FROM occurrences WHERE*
age="<>ToString[myAge]<>";"];

Extracting data for Diversity functions

Several functions in the Quantitative Paleontology package require tables of first and last appearances (*DiversityPlot[], DiversityTable[], CohortAnalysis[]*). The range data for these must be organized like the table below, with taxon name in the first column, a dummy variable in the second column (which serves as a placeholder for the ancestor ID), and the first and last appearance datums in the third and fourth columns:

Taxon Name	0	FAD	LAD
Hesperocyon	0	42.45	31.05
Osbornodon	0	33.55	15.1
Mesocyon	0	33.2	24.4
Cynodesmus	0	33.1	24.4
Oxetocyon	0	33.	31.15

The column headers should not be included when the data are passed to the diversity functions.

These data can be extracted from a MySQL database of occurrences (such as a table of occurrences downloaded from the Paleobiology Database or the NOW Database). The following code finds the

FADs and LADs for each genus in the “occurrences” table and returns them to the variable *MyRanges* in the format required by the diversity functions:

```
conn = OpenSQLConnection["mySQL", Catalog -> "Felidae"];
MyRanges = SQLExecute[conn,  "SELECT Genus, 0 ,
    Max(interval_midpoint),Min(interval_midpoint) FROM occurrences WHERE
    interval_midpoint IS NOT NULL GROUP BY Genus;"];
CloseSQLConnection[conn];
```

In this example, the absolute age of each occurrence in the *occurrences* table of the *Felidae* database is stored in the field *interval_midpoint* and the genus name is stored in the field *Genus*.

Functions

CohortAnalysis[data, censuspoint]

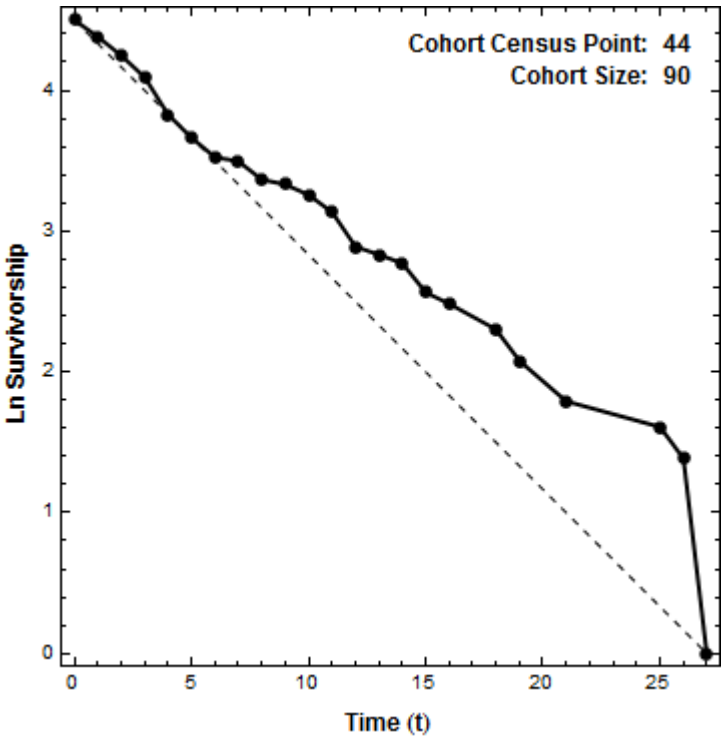
This function performs a cohort analysis on range data, either the ranges of real taxa (see "Retrieving Data from MySQL" subsection on "Extracting data for diversity functions") or simulated ranges from the *LetThereBeLight[]* function.

data – a variable containing a table of parameters and a table of stratigraphic ranges.
censuspoint - the point in time from which the cohort is taken.

Example: The following command does a cohort analysis by sampling the ranges stored in the variable *MyCreation* at time 20:

CohortAnalysis[***MyCreation, 20*** *]*

The output is graphical, showing the natural log of the number of members of the cohort remaining at time *t* as a thick line with a point at each *t* where one or more lineages became extinct. The thin dashed line shows the linear decay of a perfectly time homogenous process. The census point and the number of lineages in the cohort are reported in the upper right corner of the graph.



ConvertDIVAPoints[FileName]

This function takes a list of PointID numbers as generated by the DIVA GIS "Extract from Polygons" tool. **Note**: this function is specifically designed to work with a mySQL database into which the Polly Equidistant Point data have been loaded (http://mypage.iu.edu/~pdpolly/Data.html).

> ***FileName*** – the name of the file containing the point GlobalIDs (with path, if necessary) in double quotation marks.

Example The following command reads in a text file exported from DIVA that contains the GlobalID of points in the range of the Kit Fox, *Vulpes macrotis*, and queries the mySQL EquidistantPoints database for the longitude and latitude of each one and stores the results in the variable *KitFox*:

```
KitFox = ConvertDIVAPoints["C:\\Data\\vulpesmacropuspoints.txt"];
```

The form of the resulting data is:

GlobalID	Longitude	Latitude
133874	-112.24	43.7368
133875	-111.618	43.7368
134443	-119.53	43.2876
134444	-118.913	43.2876
134454	-112.743	43.2876

These data can be exported for use in DIVA by putting the name of the file to be created, the variable containing the data (*KitFox*, in this case) and "CSV" (to export as comma delimited text) into the command *Export[]*:

```
Export["C:\\ Data\\KitFoxCoordinates.csv", KitFox, "CSV"]
```

Or they can be mapped in Mathematica:

```
Graphics[{{Gray, CountryData["World", "Polygon"]},  Point[#] & /@ KitFox[[2 ;;, 2 ;; 3]]}]
```

Note that the above command takes columns 2 and 3 from rows 2 and higher (the first row contains column names) from the variable *KitFox* (which are the Longitude and Latitude) and maps (/@) them into the command Point[], which is a graphics feature (called a "graphics primitive" in Mathematica). The resulting points are plotted using the Graphics[] function along with a world map produced by the function CountryData[].

ConvertParacladeToTree[myCreation]

This function converts the output of LetThereBeLightWithPseudoExtinction[] to a NewickTree format. Note that it will not work with output of LetThereBeLight[], because the latter function allows ancestor species to continue past the point of speciation, which is incompatible with Newick format trees.

myCreation – a table generated by LetThereBeLigthWithPseudoExtinction[].

Example: The following code creates a paraclade and then converts it to a tree.

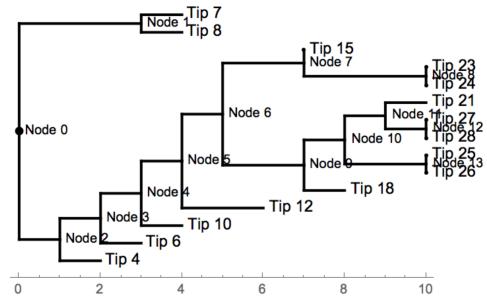
myCreation = LetThereBeLightWithPseudoExtinction[*10, 1, 0.4, 0.5*];

```
Out[345]= {{1, 1, 10, 10}, {2, 1, 10, 7}, {3, 1, 10, 9}, {4, 3, 9, 8}, {5, 3, 9, 8}, {6, 4, 8, 7}, {7, 4, 8, 7}, {8, 2, 7, 6},
          {9, 2, 7, 6}, {10, 6, 7, 6}, {11, 6, 7, 6}, {12, 10, 6, 5}, {13, 10, 6, 4}, {14, 12, 5, 3}, {15, 12, 5, 3},
          {16, 14, 3, 3}, {17, 14, 3, 0}, {18, 15, 3, 2}, {19, 15, 3, 2}, {20, 18, 2, 1}, {21, 18, 2, 0}, {22, 20, 1, 0},
          {23, 20, 1, 0}, {24, 17, 0, 0}, {25, 17, 0, 0}, {26, 21, 0, 0}, {27, 21, 0, 0}, {28, 23, 0, 0}, {29, 23, 0, 0}}
```

myTree = ConvertParacladeToTree[*myCreation*]

```
Out[346]= ((Tip 7:1,Tip 8:1):3,((((((Tip 15:0,(Tip 23:0,Tip 24:0):3):2,(((Tip 21:1,(Tip 27:0,Tip
           28:0):1):1,(Tip 25:0,Tip 26:0):2):1,Tip 18:1):2):1,Tip 12:2):1,Tip 10:1):1,Tip 6:1):1,Tip 4:1):1);
```

DrawNewickTree[*myTree*]



CreationToTree (myCreation)]

This function converts the output of LetThereBeLightWithPseudoExtinction into a Newick format tree. The function requires the package *Phylogenetics for Mathematica* to be installed (https://github.com/pdpolly/Phylogenetics-for-Mathematica).

myCreation – a table of stratigraphic ranges in the format produced by *LetThereBeLightWithPseudoExtinction[].*

Example: the converts a data set created with *LetThereBeLightWithPseudoExtinction[]* and stored in the variable *myCreation* into a Newick tree:

*mytree = CreationToTree[**myCreation**]*

Disparity[Data , (Ages, NumBins)]

This function does an analysis of disparity on a set of discrete character data. The function calculates a distance matrix, treating characters as additive (ordered), then does a principal coordinates analysis (PCO) on that matrix after it has been converted to a similarity matrix and double centered. Disparity statistics except area are based on all dimensions of the PC0, the plot and the area occupied statistics are based on the first two dimensions only. Distances are the average number of character differences between the taxa based on non-missing data. Characters are treated as unordered (i.e., distance between states 0 and 1 or between states 0 and 3 are both treated as 1 character difference). The distance between two taxa will always range between 0 and 1. For example, if two taxa share ten non-missing characters, six of which differ in state, their distance will be 0.6.

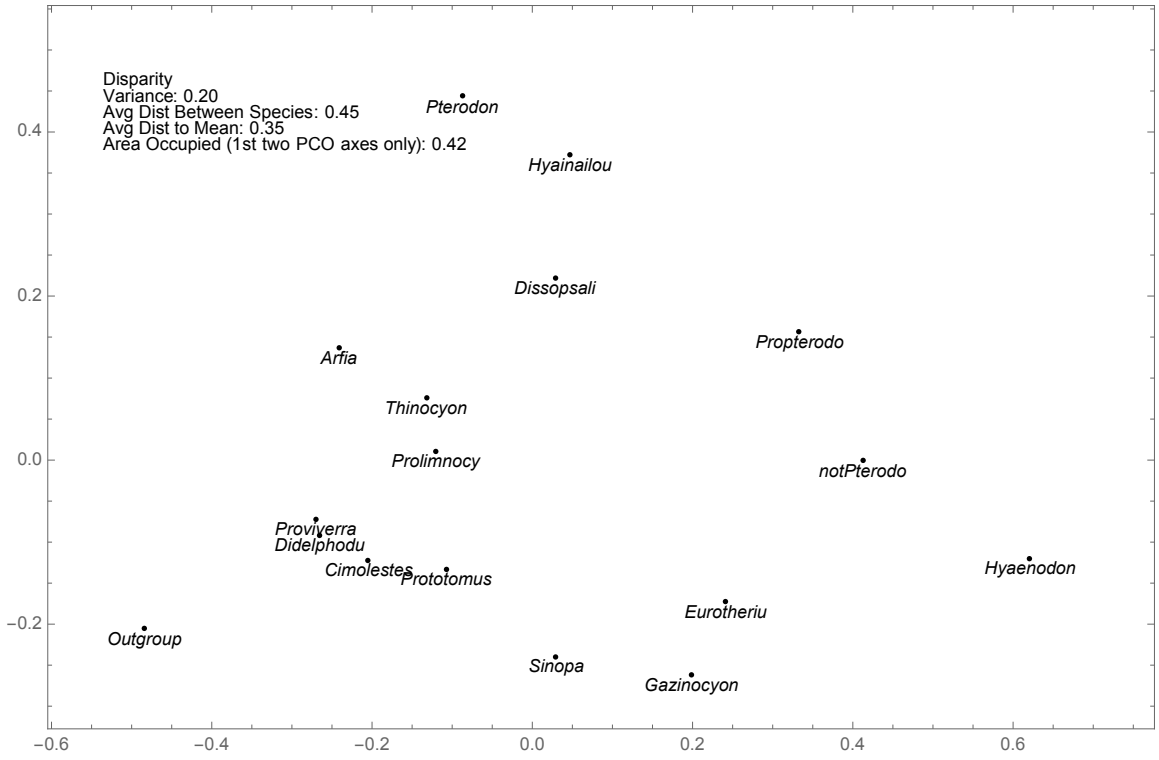
Data – a variable containing a table of character data, with taxon names in the first column. Missing data should be scored as "?" and are excluded from the calculation of distance and similarity. All other data should be integer data (0, 1, 2, etc.).

Ages – an optional argument containing the first and last appearance data (FAD and LAD) for the taxa in data, FAD in the first column, LAD in the second column with taxa in the same order as in the *Data* table. If age data are entered, the number of bins must be specified.

NumBins – an optional argument that *must* be used if the *Ages* argument is used. *NumBins* is the number of equal-interval temporal bins into which taxa should be divided based on the FAD and LAD data in *Ages*.

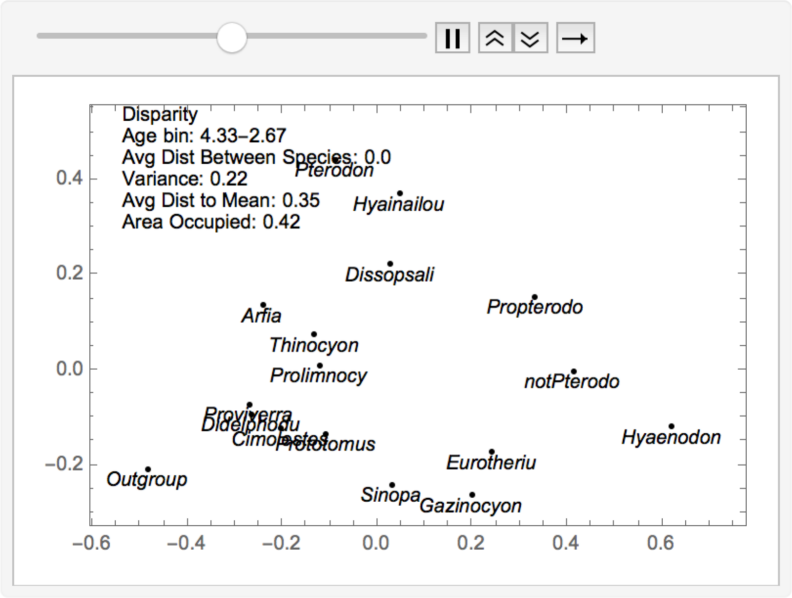
Example: The following command does a disparity analysis on the data stored in the variable *CharacterData* without age binning:

Disparity[***CharacterData***]


Disparity
Variance: 0.20
Avg Dist Between Species: 0.45
Avg Dist to Mean: 0.35
Area Occupied (1st two PCO axes only): 0.42
Pterodon
Hyainailou
Dissopsali
Propterodo
Arfia
Thinocyon
Prolimnocy
notPterodo
Proviverra
Didelphodu
Cimolestes
Prototomus
Hyaenodon
Eurotheriu
Outgroup
Sinopa
Gazinocyon
0.4
0.2
0.0
−0.2
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

The following command uses the age data options to divide the analysis into 5 separate time intervals. The output is presented as a table of graphs. These can be conveniently viewed as an animation by embedding the command in *ListAnimate[ ]*. The interval for each bin is reported in the caption to the graph.

ListAnimate[Disparity[***CharacterData, FadLadData***, 5]]



DisparityWithStages[Data , Stages, StageOrders]

This function does an analysis of disparity on a set of discrete character data. The function calculates a distance matrix, treating characters as additive (ordered), then does a principal coordinates analysis (PCO) on that matrix after it has been converted to a similarity matrix and double centered. The data are then broken down by geological age and plotted. Disparity statistics and plot are based on the first two dimensions of the PCO. Distances are the average number of character differences between the taxa based on non-missing data. Characters are treated as unordered (i.e., distance between states 0 and 1 or between states 0 and 3 are both treated as 1 character difference). The distance between two taxa will always range between 0 and 1. For example, if two taxa share ten non-missing characters, six of which differ in state, their distance will be 0.6.

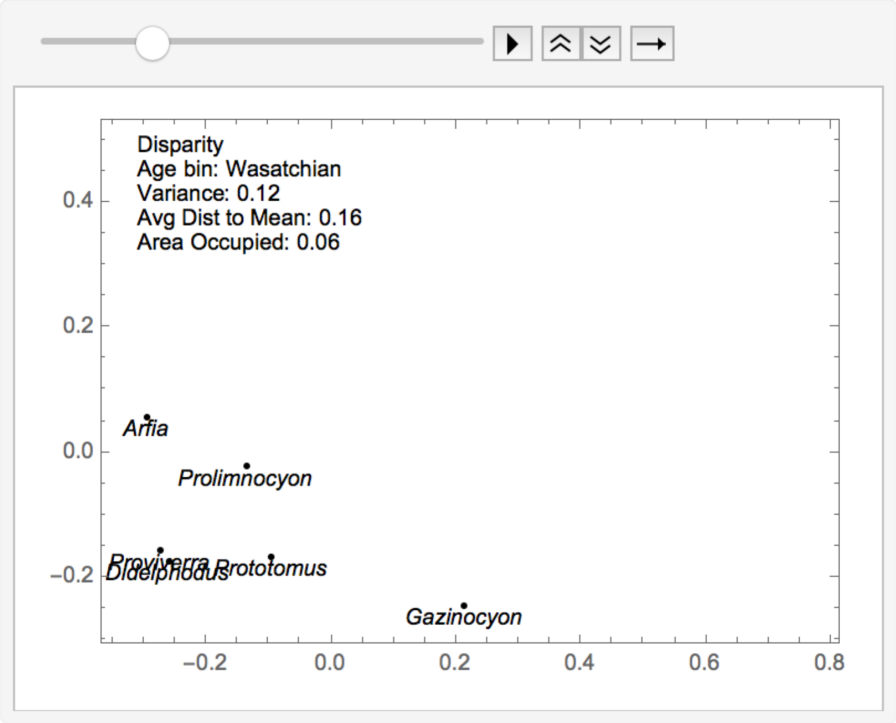
Data – a variable containing a table of character data, with taxon names in the first column. Missing data should be scored as "?".

Stages – an argument in which a two-column array containing the first and last appearance data (FAD and LAD) for the taxa in *Data*, where the FADs and LADs are the names of the geological periods in which the taxa first and last appeared.

StageOrders – an argument in which a two column array containing the names of the geological periods used in *Stages* are in the second column and their ordering from oldest to youngest is in the first column.

Example: The following command does a disparity analysis on the data stored in the variable *CharacterData,* the FADs and LADs are in *StageData*, and the ordering of the geological period names are in *StageOrders*. Using *ListAnimate[]* causes the results to be plotted as an animation that moves from the oldest to youngest period.

ListAnimate[DisparityWithStages[**CharacterData, StageData, StageOrders**]



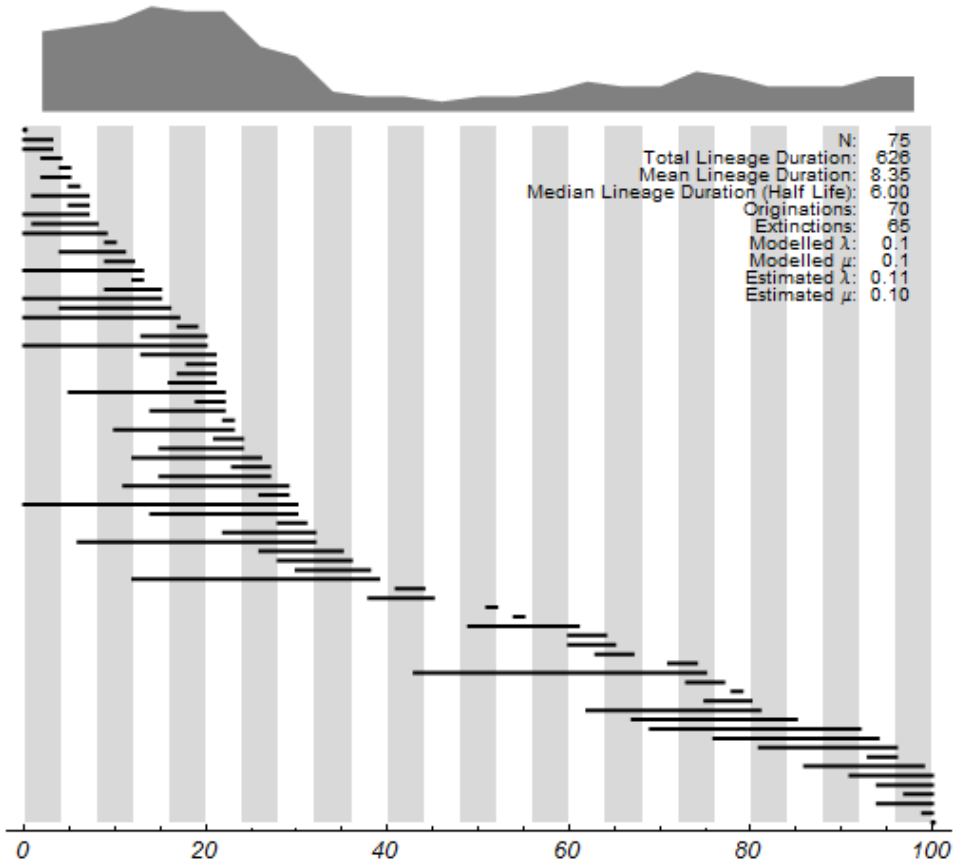
DiversityPlot[data, numbins (, λ, μ)]

This function calculates a diversity curve and plots it along with the ranges of real taxa (see "Retrieving Data from MySQL" subsection on "Extracting data for diversity functions") or simulated ranges from the *LetThereBeLight[]* function.

data – a variable containing a table of parameters and a table of stratigraphic ranges in the format produced by *GetRanges[]* or *LetThereBeLight[].*
numbins - the number of bins for the diversity analysis.
λ – the rate of origination used to model ranges in *LetThereBeLight[]*. This parameter should not be used with real data.
μ – the rate of extinction used to model ranges in *LetThereBeLight[]*. This parameter should not be used with real data.

Example: the following analyzes diversity in a data set created with *LetThereBeLight[]* and stored in the variable *MyCreation* using 25 time bins:

*DiversityPlot[**MyCreation, 25, 0.1, 0.1**]*



The resulting plot shows the ranges of all the lineages in the data set. In the upper right corner the following are reported: the total number of lineages (*N*), the sum of the durations of all *N* lineages (*Total Lineage Duration*), the mean and median lineage durations (the latter is the half-life of the clade, or the time it takes for half of the lineages to become extinct), the number of originations and extinctions in the data set, the probabilities of origination and extinction used to simulate the data (*Modelled λ* and *Modelled μ, t*hese parameters are reported as "NA" when real data are analyzed), and estimates of the origination and extinction probabilities made from the data themselves (*Estimated λ* and *Estimated μ*). In theory the modeled and estimated values should be the same, but in practice they differ by chance.

The curve at the top of the plot shows changes in standing diversity, estimated for the number of bins specified in the command line. The diversity curve includes singletons and is constructed using the "standard method" described in Box 8.2 of Foote and Miller (2006).

DiversityTable[data, numbins]

This function reports the diversity curve data in tabular form the ranges of real taxa (see "Retrieving Data from MySQL" subsection on "Extracting data for diversity functions") or simulated ranges from the *LetThereBeLight[]* function.

data – a variable containing a table of parameters and a table of first and last appearance datums in the format produced by *LetThereBeLight[]*.

numbins - the number of bins for the diversity analysis.

Example: The following command produces a table of diversity counts for the data stored in the variable *Ranges* using 5 equal-length time bins.

DiversityTable[***Ranges, 5***]

The resulting table looks like the following, where the left hand column reports the mid-point age of each bin, "N orig" reports the number of originations, "N ext" reports the number of extinctions, and "Diversity" reports the diversity count (including singletons) using the standard procedure outlined in Box 8.2 of Foote and Miller (2006).

	N orig	N ext	Diversity
4.245	14.	8.	20.
12.735	11.	15.	21.
21.225	9.	13.	23.
29.715	18.	5.	19.
38.205	1.	0.	1.

ExportKML[FileName, Coordinates, Labels->labels, Descriptions->descriptions, ColorGroups->grouplabels, AgeBins->num bins, AgeData->ages]

This function reformats and exports data so that it can be displayed in GoogleEarth. The name of the file where the data are to be saved and pairs of longitude-latitude points are required. Options allow you to label the GoogleEarth markers, add descriptions to the pop-up ballons, color them by group, or divide them into colored age bin groups.

FileName – the name of a new KML file where your GoogleEarth data will be saved.
Coordinates – the longitude and latitude coordinates of all the points. These data should be in two columns, longitude in the first and latitude in the second, each row being a different point marker.
Labels – this option gives each point a label. *labels* is a list that has as many labels as there are coordinates, in the same order as the coordinates. In other words, the first label in the list will be applied to the first set of longitude-latitude coordinates.
Descriptions – this option adds extra information to the pop-up box in each GoogleEarth marker. *descriptions* should be a list in the same order as coordinate points.
ColorGroups – this options causes markers to be colored by group. *groups* is a list that is as many labels as there are coordinates. A different color will be applied to each marker according to which group label it has. Group labels won't be printed in GoogleEarth unless you use the same list in the *labels* or *descriptions* option.
AgeBins – this option, which is paired with *AgeData*, causes data to be divided into

age bins of equal duration and markers to be colored by their age group. The *Groups* option is over-ridden by *AgeBins*. *num bins* is the number of bins.
AgeData – this option provides the age data that will be used to generate the age bins. *ages* is a list of numbers representing the age of each coordinate point.

The data for *ExportKML[]* can come from anywhere, but often they will be stored in your database and must be extracted in the proper format before using the function.

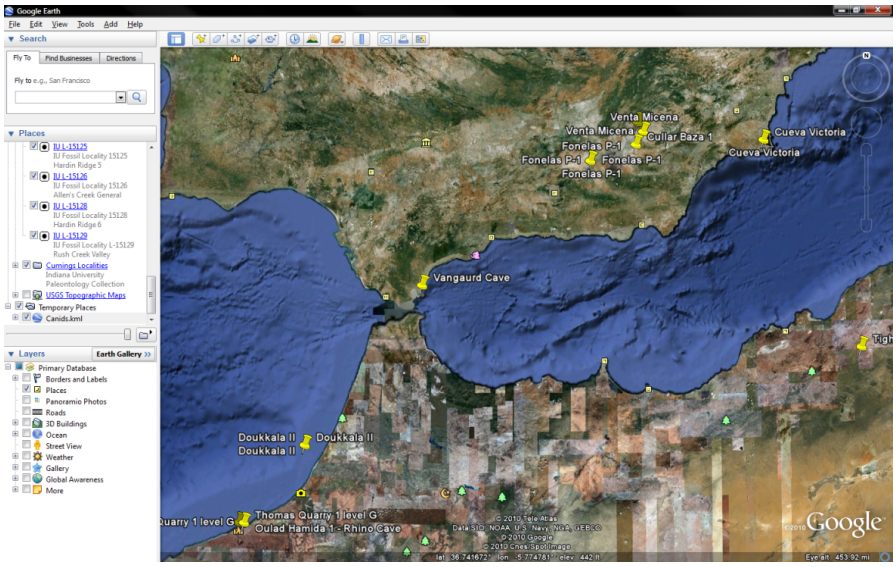
Examples The following commands retrieve the appropriate data from mySQL. Data are stored in the variable ***Results***. Note that longitude and latitude have been selected in proper order so that the first two columns contain the coordinate values. The remaining columns (Genus, Locality, AgeMidpoint) can be used as options to label or bin markers.

```
conn=OpenSQLConnection["mySQL",Catalog->"carnivorediversity"];

Results=SQLExecute[conn,"SELECT Longitude,Latitude,Genus,Locality,AgeMidpoint
FROM PBDB;"];

CloseSQLConnection[conn];
```

The following command causes simple markers to be exported, each labeled with the locality name. *Results[[1;;, 1;;2]]* is the first two columns of *Results*, which contain the longitude latitude coordinates, and *Results[[1;;, 4]]* is the fourth column of *Results*, which contains the locality names.

```
ExportKML["C:\\Desktop\\carnivores.kml", Results[[1 ;;, 1 ;; 2]],
Labels -> Results[[1 ;;, 4]] ]
```

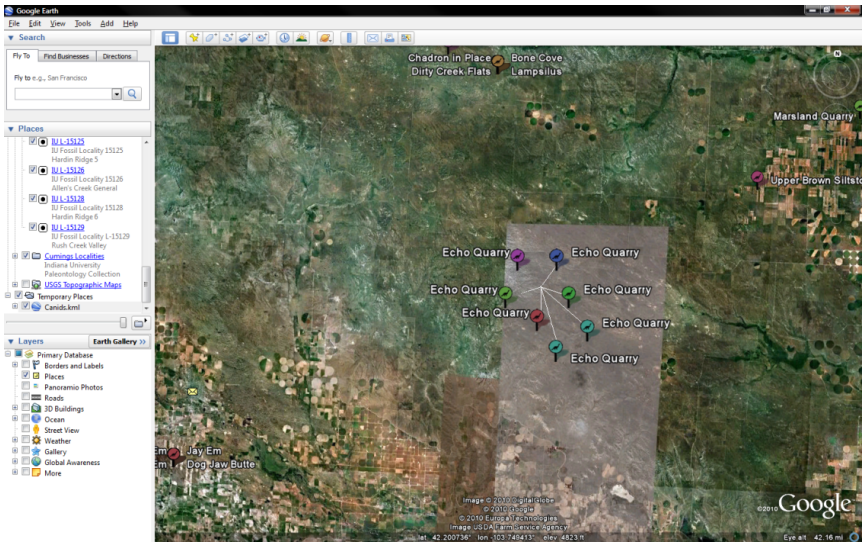
When the resulting file ("carnivores.kml") is displayed in GoogleEarth, the results look like this:



The following command causes markers to be exported, each labeled with the locality name, and each colored according to genus with a pop-up description that gives the genus name.

ExportKML["C:\\Desktop\\carnivores.kml", **Results[[1 ;;, 1 ;; 2]]**,
Labels -> **Results[[1 ;;, 4]]** , Descriptions->**Results[[1;;,3]]**,
ColorGroups->**Results[[1;;,3]]** *]*

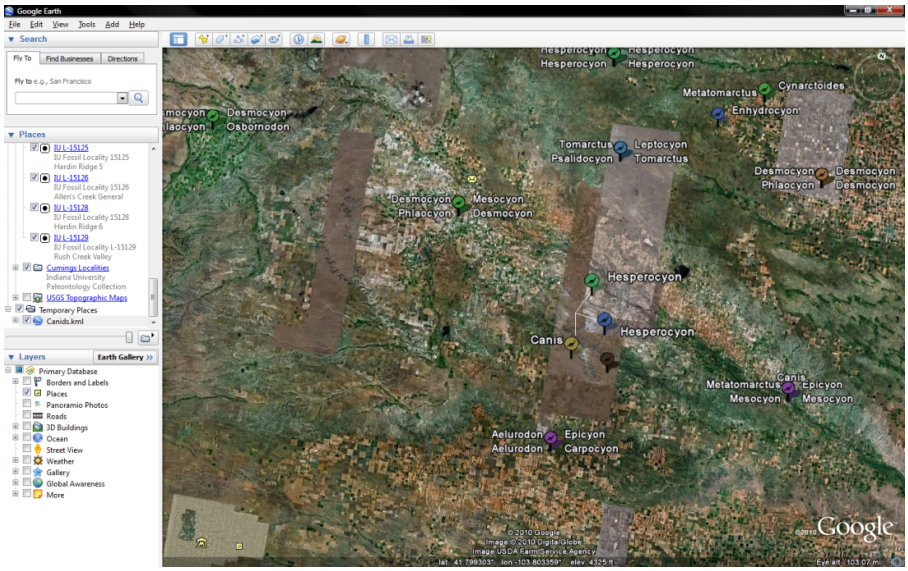
When the resulting file is viewed in GoogleEarth, the points look like this:



Finally, the following command causes markers to be exported, each labeled with the genus name, and each colored according to one of ten age bins with a pop-up description that gives the age.

ExportKML["C:\\Desktop\\carnivores.kml", **Results[[1 ;;, 1 ;; 2]]**,
Labels -> **Results[[1 ;;, 3]]** , AgeBins->**10**, AgeData->**Results[[1;;,5]]** *]*

When the resulting file is viewed in GoogleEarth, the points look like this:



GowerDistance[data]

The *GowerDistance[]* function returns a square distance matrix based on Gower's method (Gower, 1966; Legendre & Legendre, 1978). Gower distances are used when variables have incommensurate or arbitrary units (e.g., a combination of angles and length measurements or a cladistics phylogenetic data set of integer coded characters). This function handles missing data, which should be coded with a "?". This function is called automatically in Principal Coordinates Analysis when the "Gower" option is used. All variables are treated as ordered.

data – a rectangular matrix with objects in rows and variables in columns (missing data should be coded with "?")

Example:

```
dist = GowerDistance[data];
```

LetThereBeLight[time, a, λ, μ (, maxprotect)]

The *LetThereBeLight[]* function does a simulation of the evolution of a paraclade (*sensu* Raup, 1985).

time – the duration of the simulation
a – the number of lineages as the start of the simulation
λ- the speciation rate
μ - the extinction rate
maxprotect - option that breaks simulation at 10,000 lineages

Example: the following command does a simulation for 100 time units, starting with 100 lineages, each with a probability of speciation of 0.2 and a probability of extinction of 0.3.

```
MyCreation = LetThereBeLight[100, 100, 0.2, 0.3];
```

In this example, the results are saved in a new variable "*MyCreation*" and the semicolon at the end stops them from being printed to the screen. Note that if the rate of origination (***λ***) is higher than the rate of extinction(***μ***), a VERY large output may result!

The simulation is time homogenous in that each lineage has the same probability of speciation and extinction throughout the simulation. The simulation proceeds step-by-step for the number of intervals that you specify with ***time***. In other words, if you enter ***1*** the simulation checks once for speciation and extinction then stops.

By default, the function breaks early if standing diversity reaches 10,000 lineages. Setting the optional fifth parameter to FALSE allows the function to continue for the full number of steps specified by ***time***. Note that if speciation greatly exceeds extinction and if there are many steps in the simulation, the number of lineages generated could exceed memory capacity of your machine.

LetThereBeLight[] returns a range chart of all the taxa in the simulation with the following format (column labels are not included in the output):

Taxon ID	Ancestor ID	FAD	LAD
1	1	100	54
2	2	100	98
3	2	99	86
4	1	95	89
5	3	93	89

The output of *LetThereBeLight[]* can be plotted and analyzed with functions such as *DiversityPlot[]*, *DiversityTable[]*, and *CohortAnalysis[]*.

LetThereBeLightWithPseudoExtinction [time, a, λ, μ (, maxprotect)]

The *LetThereBeLightWithPseudoExtinction[]* function performs a simulation of the evolution of a paraclade (*sensu* Raup, 1985) exactly like *LetThereBeLight[]* except that when speciation occurs, the parent lineage ends and two daughter lineages begin. The addition of such pseudo extinctions will inflate the rates of extinction and speciation estimated from the data.

time – the duration of the simulation
a – the number of lineages as the start of the simulation
λ- the speciation rate
μ - the extinction rate
maxprotect - option that breaks simulation at 10,000 lineages

Example: the following command does a simulation for 100 time units, starting with 100 lineages, each with a probability of speciation of 0.2 and a probability of extinction of 0.3.

*MyCreation = LetThereBeLightWithPseudoExtinction[**100, 100, 0.2, 0.3**];*

In this example, the results are saved in a new variable "*MyCreation*" and the semicolon at the end stops them from being printed to the screen. Note that if the rate of origination (***λ***) is higher than the rate of extinction(***μ***), a VERY large output may result!

The simulation is time homogenous in that each lineage has the same probability of speciation and extinction throughout the simulation. The simulation proceeds step-by-step for the number of intervals that you specify with ***time***. In other words, if you enter ***1*** the simulation checks once for speciation and extinction then stops.

By default, the function breaks early if standing diversity reaches 10,000 lineages. Setting the optional fifth parameter to FALSE allows the function to continue for the full number of steps specified by ***time***. Note that if speciation greatly exceeds extinction and if there are many steps in the simulation, the number of lineages generated could exceed memory capacity of your machine.

LetThereBeLightWithPseudoExtinction[] returns a range chart of all the taxa in the simulation with the following format (column labels are not included in the output):

Taxon ID	Ancestor ID	FAD	LAD
1	1	100	54
2	2	100	98
3	2	99	86
4	1	95	89
5	3	93	89

The output of *LetThereBeLightWithPseudoExtinction[]* can be plotted and analyzed with functions such as *DiversityPlot[]*, *DiversityTable[]*, and *CohortAnalysis[]*, but note that these functions will produce elevated estimates of extinction and speciation because of the pseudoextinction issue. The results can be plotted with DrawNewickTree[] and analysed with other functions in the *Phylogenetics for Mathematica* package.

LRI[*timeseries* (, sd)]

This function uses the Log-rate/Log-interval (LRI) method of Gingerich (1993) to estimate the step rate (the average amount of change per step in the evolutionary process) and mode of evolution of a time series of trait means taken from sequential samples from single anagenetic lineage. The method takes advantage of the expected scaling relationship between net rate (absolute trait difference between two samples divided by the interval of time separating them) and mode of evolution. Here mode refers to whether the evolutionary process is random (Brownian motion), directional, or stabilizing.

Arguments:

- *timeseries* is a list of trait data with sample age in the first column and mean trait value in the second column.
- *sd* is an optional parameters specifying the pooled standard deviation of the samples. By default this value is 1.0.
- **Note:** LRI scaling is based on standard deviates of trait change. When SD is not specified, data may fall outside the plot boundaries if the estimated step rate is too large to be in units of standard deviations per generation. For example, a real evolutionary rate of 0.25 standard deviations per generation is realistically high; a rate of 2 standard deviations per generation is improbably high.

When net rates and intervals are plotted on a base-10 log-log plot, the intercept of a regression through them provides an estimate of the step rate and the slope provides an estimate of the evolutionary mode. Under a stabilizing process, divergence does not increase with interval, so net rates get smaller as interval increases (slope = -1.0); under a directional process, divergence increases linearly with time so net rates are constant regardless of the interval over which they were measured (slope = 0.0); under a random process, net rates get smaller as interval increases, but not as much as with a stabilizing process (slope = 0.5).

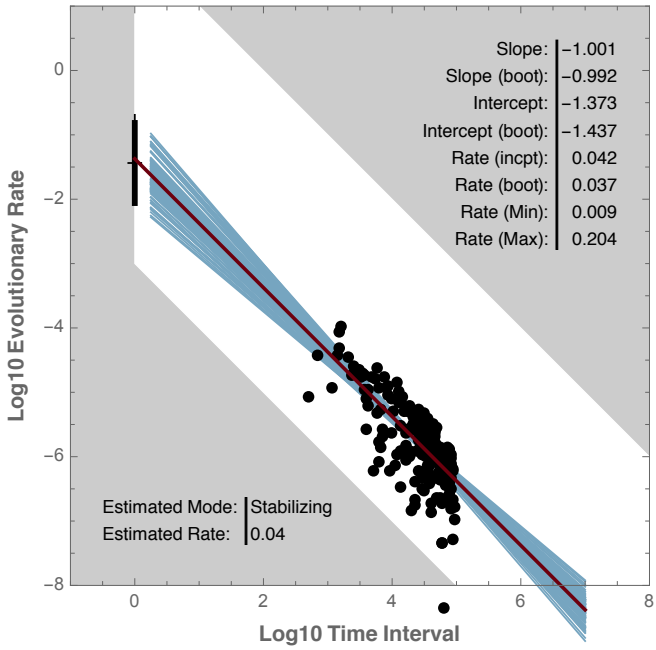
The function returns an LRI plot of net rates, a robust regression line through them, and 100 regression lines through bootstrapped data. Parameters for the slope, intercept, and estimated rate are reported in the upper right corner. Crosshairs show the best estimate of the intercept (median of the bootstrapped intercepts), the 95% confidence interval, and the full range of bootstrapped

intercepts. The best rate and the estimated mode of evolution are reported in the lower left (directional slope > -0.25, stabilizing slope < -0.75, random slope between -0.25 and -0.75).

The LRI method is described in full by Gingerich (1993; see also Gingerich 2001). The method was criticized by Sheets and Mitchell (2001). Their criticism appears to be founded on a misunderstanding of Gingerich, because they draw identical conclusions that the apparent scaling between net rates and intervals is an artifact of random walk processes and that mode of evolution must be considered when estimating rates of evolution. Hunt (2012) showed that the estimation of step rates from LRI can be biased for many ordinary paleontological data sets.

Example:

```
LRI[mydata]
```



PrincipalCoordinatesAnalysis[*data (, "Gower")*]

Performs a Principal Coordinates Analysis (PCO) and returns a matrix of PCO scores, with objects in rows and axes in columns. By default, the function uses Euclidean distances, which should always be used with geometric morphometric data and which will produce results identical to Principal Components Analysis (PCA) on the same variables. Gower distances can be substituted by adding the option "Gower". Gower distances should be used for matrices in which there are missing data or that have variables with incommensurate units (e.g., body mass, femur length, and pubic angle; or meristic data such as are found in many phylogenetic matrices). Note that Gower distance places equal weight on each variable, whereas Euclidean distances allows each variable to contribute according to its range of variation. Note also that ordinations of the same data using Euclidean and Gower distances will differ because of their differences in variable weighting.

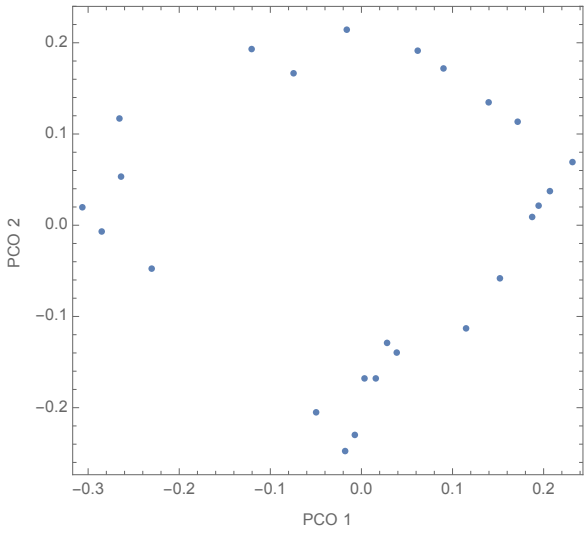
The PCO ordination produced by this function follows the algorithm described by Legendre & Legendre (1998), which scales the distance matrix by squaring it and dividing by -2, double centering, and obtaining scores from the resulting eigenvectors scaled by the square-root of the

eigen values (see also Tongerson, 1958; Gower, 1966). This method produces slightly different results from the later algorithm proposed by Mardia (1978) for non-Euclidean matrices, which is used in the popular *cmdscale()* function in R. For Euclidean matrices, higher axes often have negative eigenvalues with the Mardia method, but not with the method used here.

data – a rectangular matrix with objects in rows and variables in columns (missing data should be coded with "?")
Gower - option for Gower distance instead of default Euclidean distance

Example:

```
scores = PrincipalCoordinatesAnalysis[data, "Gower"];

ListPlot[scores[[1 ;;, {1, 2}]], AspectRatio -> Automatic,  FrameLabel -> {"PCO 1",
"PCO 2"}, Frame -> True, Axes -> False]
```



QuantitativePaleontologyVersion[]

Prints the version number and citation for the current installation.

Example:

```
QuantitativePaleontologyVersion[]

Quantitative Paleontology for Mathematica 5.1
(c) P. David Polly, 23 May 2023
```

StigallAreaAnalysis[Data]

This function calculates standardized geographic range areas for a set of taxa with longitude and latitude coordinates (Stigall 2010). The function takes all the geographic coordinates, projects them to a Lambert Azimuthal equal area projection, and calculates the areas of the convex hull surrounding the longitude-latitude points for each taxon, dividing them by the area inside the convex

hull surrounding all points (a method of standardizing for outcrop size). The area of each species thus reported as a proportion of the total area. Note that biases may be introduced by working with taxa with large geographic ranges. For example, if taxa occur in both North America and Europe the geographic area calculated by this function may include parts of the Atlantic Ocean. To recreate Stigall's analysis you would divide data into time bins, use this function to calculate areas for each time bin, and then analyze the results.

> ***Data*** – a matrix consisting of three columns: Longitude, Latitude, and Taxon name in that order. Each row is a different occurrence point.

Example The following command reads creates a data set for taxa in the "Cenozoic 6" time bin by querying the mySQL database :

```
conn=OpenSQLConnection["mySQL",Catalog->"carnivorediversity"];
Results=SQLExecute[conn,"SELECT Longitude,Latitude,Genus FROM PBDB WHERE
Age10mybin="Cenozoic 6";"];
CloseSQLConnection[conn];
```

Then the following calculates the standardized geographic range areas for those taxa:

```
Areas = StigallAreaAnalysis[Results]
```

Results look like the following:

```
  Aelurodon   0.0671123
    Alopex    0.00456838
 Archaeocyon  0.0217839
 Borophagus   0.161847
  Caedocyon       0
```

Acknowledgements

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.

References

Foote, M. and A. I. Miller. 2006. Principles of Paleontology, 3rd Edition. W. H. Freeman.

Gingerich, P. D. 1993. Quantification and comparison of evolutionary rates. *American Journal of Science*, 293-A: 453-478.

Gingerich, P. D. 2001. Rates of evolution on the time scale of the evolutionary process. *Genetica* 112-113: 127-144.

Gower, J. C. 1966. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53: 325-338.

Hunt, G. 2012. Measuring rates of phenotypic evolution and the inseparability of tempo and mode. *Paleobiology*, 38: 351-373.

Legendre, P. and L. Legendre. 1998. *Numerical Ecology, 2nd Edition*. Elsevier Science.

Mardia, K. V. 1978. Some properties of classical multi-dimensional scaling. *Communications in Statistics-Theory and Methods*, A7(13): 1233-1241.

Raup, D. M. 1985. Mathematical models of cladogenesis. *Paleobiology*, 11: 42-52.

Sheets, H. D. and C. E. Mitchell. 2001. Uncorrelated change produces the apparent dependence of evolutionary rate on interval. *Paleobiology*, 27: 429-445.

Stigall, A. 2010 Using GIS to assess the biogeographic impact of species invasions on native brachiopods during the Richmondian invasion in the type-Cincinnatian (Late Ordovician, Cincinnati Region). *Palaeontologia Electronica*, 13.1.5A: 1-19.

Tongerson, W. S. 1958. *Theory and Methods of Scaling*. Wiley, New York.

