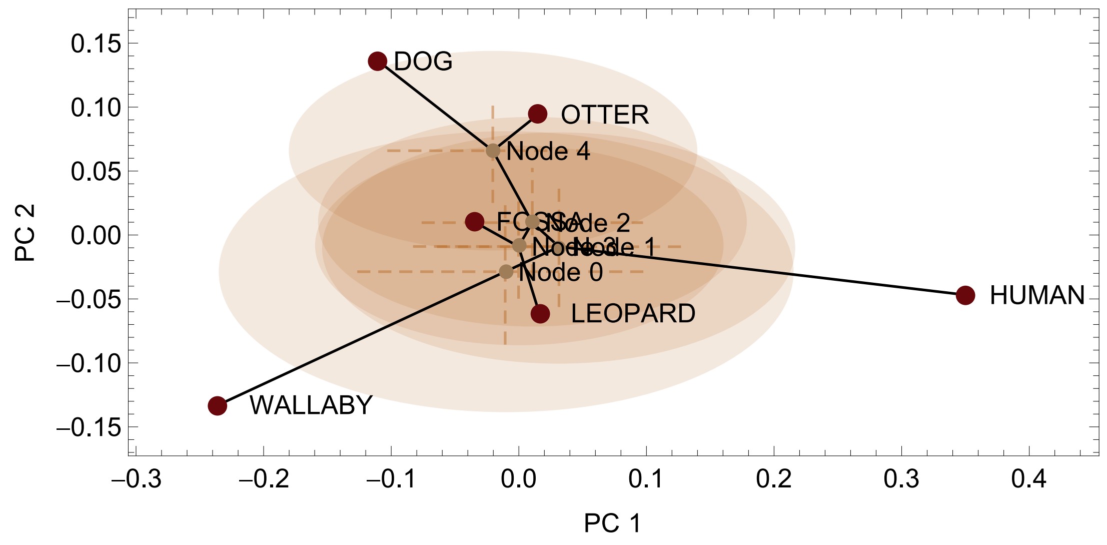
G562 Geometric Morphometrics

# *Geometric Morphometrics for Mathematica*

## User's Guide

Version 12.4, February 2022

This package is used in Indiana University course
EAS-G 562 Geometric Morphometrics



**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.  2022.  Geometric morphometrics for Mathematica. Version 12.4
Department of Earth & Atmospheric Sciences, Indiana University: Bloomington, Indiana.
https://pollylab.indiana.edu/software/

# Table of Contents

## Installing the package

1. Download the latest version of the package at http://mypage.iu.edu/~pdpolly/Software.html (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., "PollyMorphometrics")
5. Once installed, enter the command "<<PollyMorphometrics`" to use the functions.
6. Use the function *MorphometricsVersion[]* to determine which version you have installed.
7. Some functions in this package require the *Phylogenetics for Mathematica* package to be installed. The latter can be obtained from the same site.

## Changes

12.4 Fixed bug in *ShapeMANOVA[]* that misreported total sum of squares and F-ratio.
12.3 Added functions for RV and CR coefficients and their permutation tests. Converted all "Modules" to "Blocks" to conserve memory.
12.2 Added *Mantel[]* and *MantelForMorphometrics[]* tests.
12.1: Fixed bug in several functions that use reconstructed ancestral nodes, including *ReconstructAncestorShapes[]* and *PhylogeneticPrincipalComponentsAnalysis[]*. The problem was caused by a line of code in *ReconstructNodes[]* in the phylogenetics package that failed when the number of taxa was very large.
12.0: ***PrincipalCoordinates (Improved Function)***: The PrincipalCoordinates[] function returns principal coordinates scores from a data matrix based on Euclidean distance with option for Gower distance.
12.0: ***BreakOutlines* (Improved Function):** The BreakOutlines function now spaces points at equal distances across the breakpoints in addition to within the outline segments.
11.1: fixed bug in *tpsImport[]* that caused the function to fail on files without outlines.
11.2: fixed bug in *tpsSpline[]* that caused spline visualizations to be malformed in some cases. No results were affected by the original error except the shape of the spline images.
11.3: fixed bug in *BreakOutlines[]* that caused function to fail if more than one outline point was *Nearest[]* to landmark point.
11.4: fixed bug in *tpsImport[]* that caused function to fail on files with both outlines and scale factors.

## 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 youwant. 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 GMM functions

### AdamsCRTest[*proc, lands1, lands2 (,dims, permuteby)*]

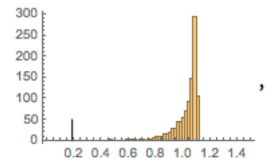
This function performs a permutation test using the CR coefficient as described by Adams (2016) to test whether two blocks of data (columns) have significantly more covariance within them than between them (i.e., whether they are "modular") and whether those same two blocks have more covariance between than within (i.e., whether they are integrated). 1,000 permutations are used in the test. The function returns the CR coefficient, a P value for whether the two blocks are modular (i.e., they covary within the block but not between the blocks), and a P value for whether the two blocks are integrated. To convert P to the probability that the observed modularity or integration could have arisen by chance, subtract it from 1.0. The function also returns a graphic that shows the random CR values from permutations as a histogram and the observed value as a vertical black line.

Arguments:

- *proc* is a matrix of 2D or 3D landmarks that have been Procrustes superimposed. Non-landmark data may be substituted, but the option *dims* must be set to 1 and *permuteby* must be set to "Variables". For 3D landmarks *dims* must be set to 3.
- *lands1* and *lands2* are numerical lists of landmarks that should be included in each block. (e.g., {1,4,5,6} and {2,3,7,8}).
- *dims* is an optional value for the number of dimensions in each landmark. By default this is 2. For non-landmark variables use 1 and for 3D landmarks use 3.
- *Permuteby* is an optional value that specifies whether permutations should keep the individual x, y, and z coordinates of each landmark together ("Landmarks") or permute them independently ("Variables").

Examples: AdamsCRTest[*proc*, {1,2,3,4}, {5,6,7,8}]

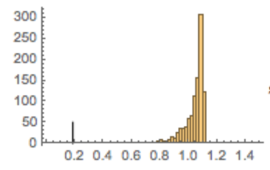
```
{CR → 0.184948, P(modular) → 1., P(integrated) → 0.,
```



```
, Permuted by landmarks.}
```

AdamsCRTest[*proc*, {1,2,3,4}, {5,6,7,8}, 3, "Variables"]

```
{CR → 0.184948, P(modular) → 1., P(integrated) → 0.,
```



```
, Permuted by columns.}
```

## BreakOutline[*outline, landmarks, {segment nums}*]

This function breaks an outline or curve into two or more segments based on the positions of one or more landmarks and places a specified number of equally spaced semilandmarks along each of the resulting segments. This procedure is based on the Extended Eigenshape procedure of MacLeod (1999) and can be used either with Eigenshape analysis proper (analysis of the covariance matrix of a *phi* function) or with Procrustes semilandmark analysis.

Arguments:

- *outline* is a matrix of 2D or 3D semilandmarks that define a closed outline or open curve. The matrix should be formatted so that the two or three coordinates of each semilandmark are partitioned together on a single row.
- *landmarks* is a matrix of 2D or 3D landmark coordinates that define one or more homologous points at which the outline curve should be broken. This matrix should also be formatted so the two or three coordinates of each landmark are partitioned in a single row. Do not include landmarks for the start and end of the original *outline* curve.
- *segment nums* is a list of integers indicating how many equally spaced semilandmarks should be placed on each of the resulting segments. Note that there will always be one more segment than there are *landmarks* (i.e., one landmark will break the outline into two segments).

Example:

```
outline = Partition[coords, 2];
landmarks = Partition[proc[[1]], 2];

BreakOutline[outline, landmarks, {50, 70}]
```

## CentroidSizes[*data*]

Returns the centroid sizes of a matrix of landmarks in which the first column contains a scaling factor stating the proportion between the units of the landmark coordinates (which are often pixels) and real units of measurement (e.g., millimeters). The function multiplies the landmarks of each object by their scaling factor, then calculates the square root of the sum of squared distances between the landmarks and their centroid.

Arguments:

- *data* a matrix of landmarks with scaling factors in the first column

Example (where *data* contains landmark configurations of six objects):

```
CentroidSizes[data]

{12.6565, 12.8118, 12.5415, 12.8007, 12.2841, 12.5147}
```

## CommonOrientation[*data1, data2, n, k*]

This function rotates one set of Procrustes aligned coordinates to the same orientation as another to make them comparable for statistical analysis when they have been aligned separately. The function returns *data2* rotated into an alignment that matches *data1*.

Arguments:

- *data1* is a matrix of Procrustes superimposed landmarks
- *data2* is another matrix of the same number of Procrustes superimposed landmarks
- *n* is the number of landmarks
- *k* is the number of dimensions of each landmark (2 or 3)

Example (where *data* contains landmark configurations of six objects):

```
newdata = CommonOrientation[data1, data2, 9, 2];
```

## CRCoefficient[*data1, data2*]

This function returns the CR coefficient of Adams (2016) for two blocks of data.

Arguments:

- *data1* is a subset of columns representing one putative module selected from a matrix of Procrustes superimposed landmarks
- *data2* is a complementary subset of columns representing a second putative module selected from a matrix of Procrustes superimposed landmarks

Example:

```
CRCoefficient[proc[[1;;,{1,2,3,4}]], proc[[1;;, {5,6,7,8}]]]

0.0183
```

## EqualSpace[*points, n (, dim, polyorder)*]

This function uses polynomial interpolation to find n equally spaced points from an input curve of x - y coordinates. Note that it only works on one object at a time. Point coordinates are returned as a simple list.

Arguments:

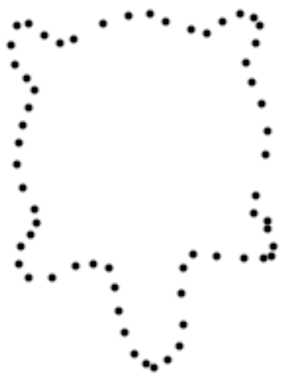
- *points* is a list of xy coordinates specifying a closed curve (outline) of a single object. The coordinates may be in a single line or may be grouped in xy pairs.
- *n* is the number of output points desired.

- *dim* is an optional parameter specifying whether the morphometric data are 2D or 3D. By default this is set to 2D. Enter "3" as an option if you have 3D data.
- *polyorder* is an optional parameter specifying the order of polynomial to be used in the interpolation. By default this is set to 1, which causes the interpolated points to be placed on a straight segments between the original points. Setting this parameter to 2 or 3 will smooth sharp bends in the original data, but could result in unexpected departures. Comparing the equally spaced points to the original data is highly recommended when choosing the setting.

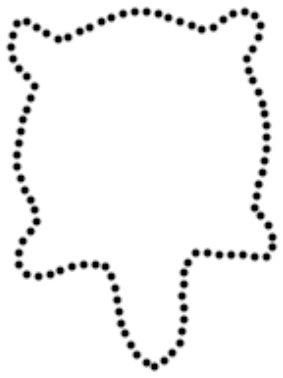
Example:

*equalpts = EqualSpace[pts, 100]];*

Original points:



Equally spaced points:



### FlipImageJ[*data*]

This function flips landmarks vertically. It is designed for use with landmarks that were collected with ImageJ, a package in which the y-axis gets larger toward the bottom of the screen rather than the top. This function is designed specially for use with the *tpsImport[]* function in this package and expects that the first column of data contains object labels and the remaining columns contain x-y landmark coordinates. Data are returned in the same form.

Arguments:

- *data* is a rectangular matrix with a column of object labels followed by columns of the x-y coordinates of several landmarks.

Example:

*data = FlipImageJ[ tpsImport[ "mandibles.tps"] ];*

Note that if you use FlipImageJ with coordinates collected from other programs, such as *tpsDig*, your landmarks will be flipped vertically.

### ImportImageJOutlines[*path, filter*]

This function imports outline coordinates created using the polygon tool and save XY coordinates function of ImageJ. The function reads in all files that match the filter, assuming that each one is a plain text file. The file name is prepended to each row of coordinates as a label. Coordinates are returned in rows with the file name in the first column.

Arguments:

- *path* is the path to the directory where the coordinate files are stored.
- *filter* matches file names and has the form "*.txt" or similar.

Example:

*ImportImageJOutlines["/Documents/Data/ImageJ/", "*.txt"];*

### ImportImageJPoints[*path, filter*]

This function imports landmarks coordinates created using the PointPicker tool of ImageJ. The function reads in all files that match the filter, assuming that each one is a plain text file. The file name is prepended to each row of coordinates as a label. Coordinates are returned in rows with the file name in the first column.

Arguments:

- *path* is the path to the directory where the coordinate files are stored.
- *filter* matches file names and has the form "*.txt" or similar.

Example:

*ImportImageJPoints["/Documents/Data/ImageJ/", "*.txt"];*

## Mantel[*matrix1,matrix 2, n*]

This function performs a Mantel Test on two matrices. It returns the matrix correlation and the probability (p) that the two have different structures (a p-value below the test's alpha level indicates the two matrices are more similar than expected by chance). The function incorporates the recommendations of Manly (1986).

**Note:** For geometric morphometric data use the *MantelForMorphometrics[]* function instead of this one.

Arguments:

- *matrix1* and *matrix2* are the two symmetric matrices whose Mantel correlation is to be calculated.
- *n* is the number of randomizations used to calculate the test statistics.

Example:

*Mantel[P, G, 5000]*

## MantelForMorphometrics[*matrix1,matrix 2, n*]

This function performs a Mantel Test on two geometric morphometric covariance matrices. It returns the matrix correlation and the probability (p) that the two have different structures (a p-value below the test's alpha level indicates the two matrices are more similar than expected by chance). The function incorporates the recommendations of Manly (1986). It differs from the *Mantel[]* function by incorporating the diagonal elements (coordinate variances) in the matrix correlation (Klingenberg and MacIntyre, 1998).

**Note:** For geometric morphometric data use the *MantelForMorphometrics[]* function instead of this one.

Arguments:

- *matrix1* and *matrix2* are the two symmetric matrices whose Mantel correlation is to be calculated.
- *n* is the number of randomizations used to calculate the test statistics.

Example:

*Mantel[P, G, 5000]*

## MorphometricsVersion[]

Prints the version number and citation for the current installation.

Example:

```
MorphometricsVersion[]

PollyMorphometrics 12.3
(c) P. David Polly, 23 March 2019
```

## PrincipalCoordinates[*data, (Gower)*]

This function returns Principal Coordinates scores for a data matrix of variables using Gower's (1966) method.

Arguments:

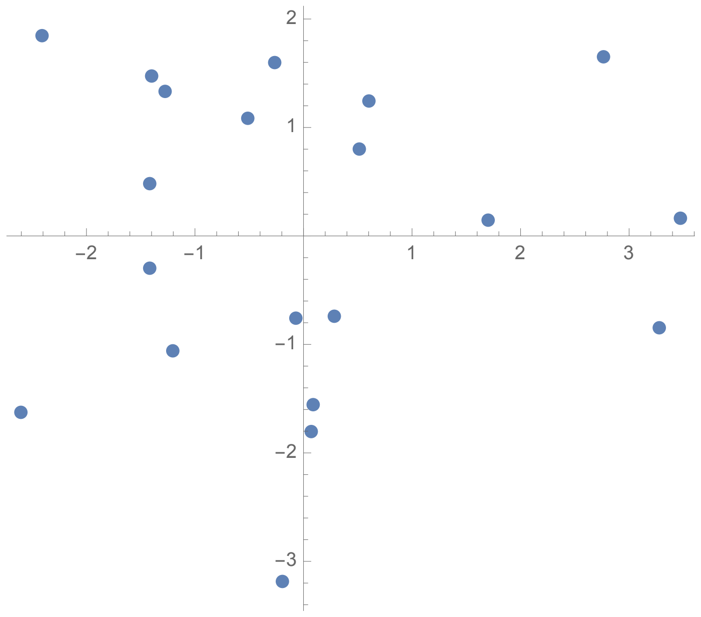
- *data* is a rectangular matrix of numeric variables
- *Gower* is an optional variable, which if set to *True* uses Gower distances instead of Euclidean distances for the PCO

The function calculates Euclidean (or Gower) distances, multiplies them by -0.5, double-centers them, and calculates the principal coordinate scores using Gower's method.

Example:

```
PCOscores = PrincipalCoordinates[data];

ListPlot[PCOscores[[1;;,{1,2}]], AspectRatio->Automatic]
```



## Procrustes[*data, n, k*]

This function performs a Procrustes superimposition of landmark coordinates followed by an orthogonal projection into tangent space. The function uses the algorithm presented by Rohlf and Slice (1990) and the tangent space projection presented by Rohlf (1999). Function works for 2D and 3D coordinate data.

Arguments:

- *data* is a rectangular matrix of landmark coordinates to be aligned, each shape in a single row and the x,y (,z) coordinates of the *n* landmarks in the columns
- *n* is the number of landmarks
- *k* is the number of dimensions of each landmark (2 or 3)

The function returns the aligned coordinates in a rectangular matrix of the same format. For 2D landmarks, the aligned shapes are returned in the same orientation as the first shape in the pre-aligned data.

Example:

```
data = {{65., 283., 100., 309., 138., 320., 170., 324.}, {77., 265., 101.,
289., 142., 301., 172., 306.}, {38., 303., 75., 331., 114., 340.,   149., 346.},
{86., 291., 114., 314., 154., 323., 186., 330.}, {88.,  462., 107., 485., 154.,
489., 193., 494.}};

aligned = Procrustes[data, 12, 2];
```

## ProcrustesDistance[*shape1, shape2, k*]

This function calculates the Procrustes distance between two shapes after it places them in optimal alignment using Procrustes superimposition. The Procrustes distances is the square root of the sum of squared distances between corresponding landmarks.

Arguments:

- *shape1* is a set of landmarks
- *shape2* is a set of landmarks from the same scheme as shape1
- *k* is the number of dimensions of each landmark (2 or 3)

Example:

```
ProcrustesDistance[landmarks[[1]], landmarks[[2]], 2]

0.0350166
```

## ProcrustesPlot[*data*]

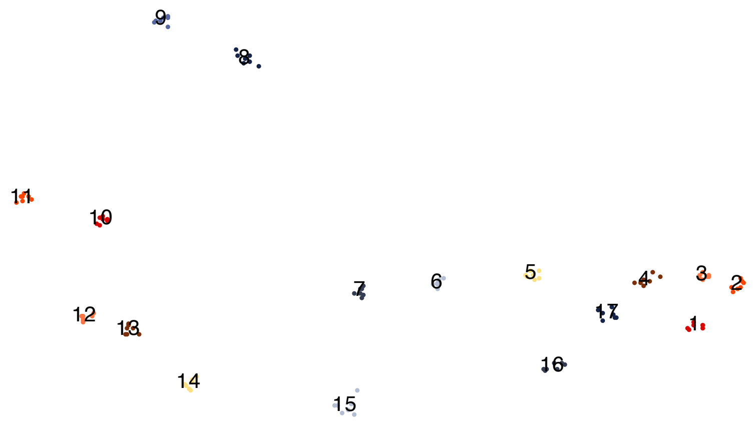
This function plots two-dimensional Procrustes aligned coordinates for illustrative purposes. Each landmark is given an identifying color and the landmarks are numbered at the corresponding centroid

Arguments:

- *data* is a rectangular matrix of 2D Procrustes aligned landmark coordinates

Example:

*ProcrustesPlot[data]*



## ProcrustesPlot3D[*data*]

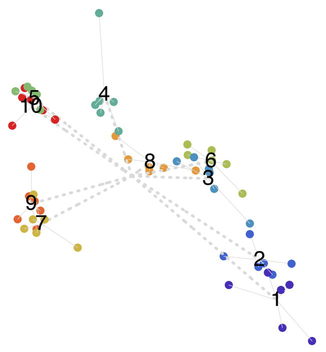
This function plots three-dimensional Procrustes aligned coordinates for illustrative purposes. Each landmark is given an identifying color and the landmarks are numbered at the corresponding centroid. Landmarks of the objects are connected to the corresponding landmark centroids by solid gray lines and each landmark centroid is connected to the object centroid by dotted gray lines.

Arguments:

- *data* is a rectangular matrix of 3D Procrustes aligned landmark coordinates

Example:

*ProcrustesPlot3D[data]*



## ProcrustesSized[*data, scalefactors, n, k*]

This function performs a Procrustes superimposition of landmark coordinates followed by an orthogonal projection into tangent space, retaining the size of the original objects. The function uses the algorithm presented by Rohlf and Slice (1990) and the tangent space projection presented by Rohlf (1999). Function works for 2D and 3D coordinate data.

Arguments:

- *data* is a rectangular matrix of landmark coordinates to be aligned, each shape in a single row and the x,y (,z) coordinates of the *n* landmarks in the columns
- *scalefactors* is a list of scaling factors between the coordinates and the original size of the objects. If the coordinate scales are in absolute units then the elements of this list will all be 1.
- *n* is the number of landmarks
- *k* is the number of dimensions of each landmark (2 or 3)

The function returns the aligned coordinates in their original scale in a rectangular matrix of the same format. For 2D landmarks, the aligned shapes are returned in the same orientation as the first shape in the pre-aligned data.

Example:

*data = {{65., 283., 100., 309., 138., 320., 170., 324.}, {77., 265., 101., 289., 142., 301., 172., 306.}, {38., 303., 75., 331., 114., 340., 149., 346.}, {86., 291., 114., 314., 154., 323., 186., 330.}, {88., 462., 107., 485., 154., 489., 193., 494.}};*

*scalefactors = { 0.023247, 0.0238, 0.019999, 0.024038, 0.024152};*

*aligned = ProcrustesSized[data, scalefactors, 4, 2];*

## ReplaceMissingWithBilateral[*object, midline, bilaterals*]

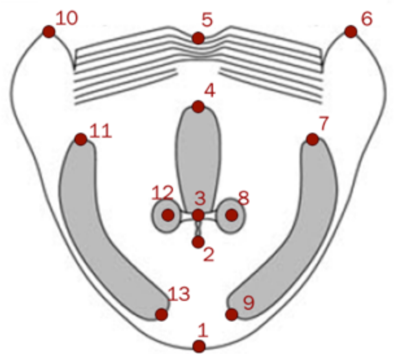
This function replaces missing landmarks by flipping their bilateral counterpart. It works on 2D data from bilaterally symmetric objects where there are at least two landmarks on the midline. Missing landmarks are coded with {"NA", "NA"}.

Arguments:

- *object* is a table of landmark coordinates with landmarks in the rows and x and y coordinates in the columns. The x and y columns of missing landmarks are filled with the string "NA".
- *midline* is a list containing the number of two midline landmarks.
- *bilaterals* is a list containing the numbers of pairs of bilaterally symmetric landmarks.

Example:

Consider the following placoderm skull with 13 landmarks. If landmarks 7 and 9 are missing, the function will replace them with 11 and 13. The function requires two midline landmarks to be specified (e.g., 1 and 5) and all pairs of bilaterally homologous landmarks to be listed.



*ReplaceMissingWithBilateral[object, {1,5}, {{6,10},{7,11},{8,12},{9,13}}]*

## RotateShape[*data, degrees*]

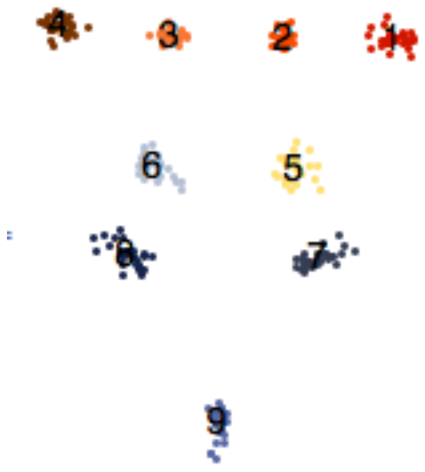
This function rotates two-dimensional shapes by a specified number of degrees. Input consists of a rectangular matrix with coordinates in the columns and objects in rows, such as the matrix returned by the *Procrustes[]* function and the number of counterclockwise degrees that the shapes should be rotated.

Arguments:

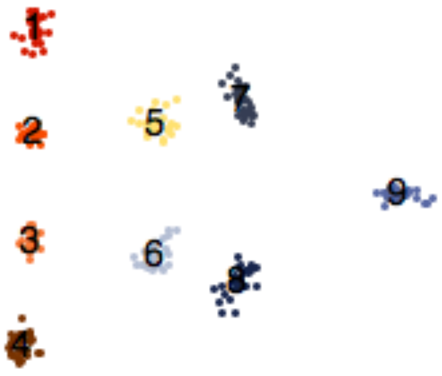
- *data* is a rectangular matrix of landmark coordinates to be aligned, each shape in a single row and the x,y (,z) coordinates of the *n* landmarks in the columns
- *degrees* is the number of degrees to rotate.

Example:

*ProcrustesPlot[proc]*



*ProcrustesPlot[RotateShape[proc,90]]*



## RVCoefficient[*data1, data2*]

This function returns the RV coefficient of Escouffier (1973) and Klingenberg (2009) for two blocks of data.

Arguments:

- *data1* is a subset of columns representing one putative module selected from a matrix of Procrustes superimposed landmarks
- *data2* is a complementary subset of columns representing a second putative module selected from a matrix of Procrustes superimposed landmarks

Example:

```
RVCoefficient[proc[[1;;,{1,2,3,4}]], proc[[1;;, {5,6,7,8}]]]

0.0183
```

## tpSpline[*source, target (, label, gridscale)*]

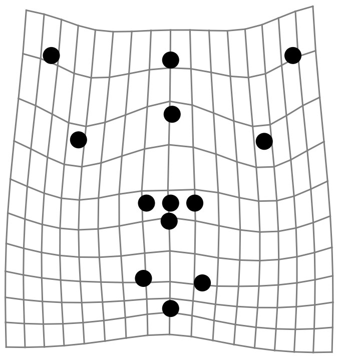
This function draws a thin plate spline deformation grid that warps the landmarks of a 2D source shape into the target shape. The shapes should be Procrustes superimposed. The algorithm is from Dryden and Mardia (1998) and Hammer and Harper (2006).

Arguments:

- *source* is a set of 2D Procrustes superimposed landmark coordinates, often the mean shape of a sample.
- *target* is another set of 2D Procrustes superimposed landmark coordinates.
- *label* is an optional string that is used as the plot label.
- *gridscale* is an optional integer that adjusts the density of grid lines in the spline plot. By default there are 10 lines on the smallest side. Set this parameter to a larger or smaller number than 10 to adjust the density.

Example:

```
source = Mean[Procrustes[data, 13, 2]];
target = Procrustes[data, 13, 2][[1]];
tpSpline[source, target]
```



## tpsImport[*filename (, imagelabels)*]

This function imports at TPS file, such as the ones used in the TPS-series of programs written by F. James Rohlf. The function recognizes both 2D and 3D landmarks. It also distinguishes between landmarks and outlines.

If the TPS file contains landmarks but no outlines then the function returns landmark coordinates with each specimen in a single row. The first column contains a specimen label that is based on the file name in the ID tag of the tps file unless "IMAGE" is specified as an option or a list of user-supplied labels is given. If a scale has been set, as with tpsDIG, the second column contains the scaling factor. Subsequent columns contain the x, y (,z) coordinates of the landmarks.

If the TPS file contains outlines, then these are returned in a second data block (multiple blocks if there is more than one outline or curve per object). Each outline block has object labels in the first column and scale factors in the second column if appropriate.

**Important Changes**: This function has been substantially changed since version 10.x. For simple 2D landmark files its behavior is unchanged, but for 3D landmarks or outline files the behavior is different. The previous version of the function is available as *tpsImportOrig[]*.

Arguments:

- *filename* is the name of the file to be imported (often with complete path).
- *imagelabels* is an optional argument that specifies the source for the object labels. By default the labels are taken from the ID tags in the TPS file, but if the option “IMAGE” is given then the labels are taken from the IMAGE tags. The user may also supply a list of labels in string format. If the length of the label list is different from the number of objects then labels default to ID tags.

Example:

*data = tpsImport[“mandibles.tps”];*

## Functions for Complete Analyses

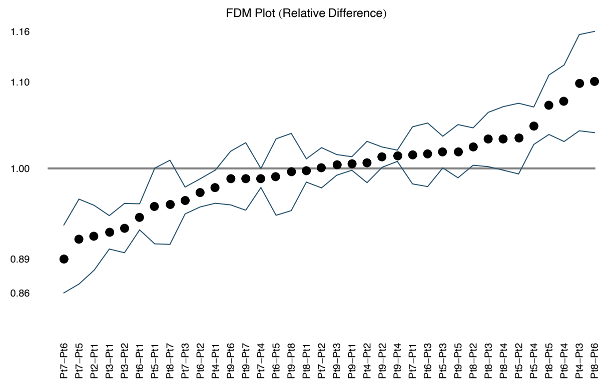
### EDMA[*landmarks1, landmarks2, landmarklabels*]

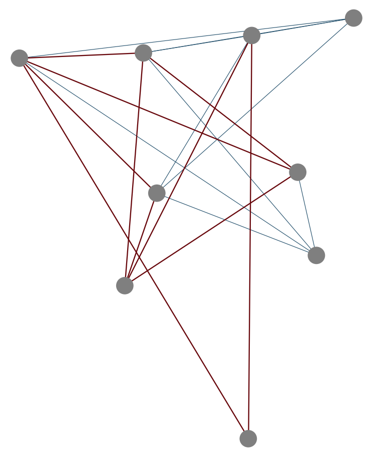
This function does a Euclidean Distance Matrix Analysis (EDMA) on two samples of two - dimensional landmark coordinates. Normally the coordinates used in EDMA are scaled in real units and have not been Procrustes superimposed. This function returns two graphs, the first showing the values of the Form Distance Matrix (FDM) of sample 2 compared to sample 1, and the second showing the mean shape of sample 2 with the lengthened interlandmark distances colored deep red and the shortened distance colored blue. The FDM is based on the relative distance method (Lele and Richtsmeier, 2001) and shows the ratio between each interlandmark distance as a black dot with a 95% confidence interval based on 1000 bootstrap resamplings of the original data (Lele and Richtsmeier, 1995). The second graphic shows those interlandmark distances whose 95% confidence intervals do not encompass 1.0 (i.e., those whose interlandmark distances are significantly different between the two samples).

Arguments:

- *landmarks1 and landmarks2* are matrices of xy coordinates for two samples, normally with the coordinates in scaled units and not Procrustes superimposed.
- *landmarklabels* is a list of landmark labels as strings.

Example: EDMA analysis of nine landmarks on the faces of people looking to the right and to the left. Landmarks show the corners of the left and right eyes, the edges of the nose, the corners of the mouth, and the chin. The transformation of faces looking right to ones looking left involves lengthening of interlandmark distances on the left side of the face (red) and shortening of those on the right side (blue).





## PhylogeneticPrincipalComponentsOfShape[*proc, labels, PCs, tree*]

This function performs a phylogenetic principal components of shape using the techniques described by Revell (2009). The function also projects the phylogenetic tree into the pPCA space using the techniques described by Rohlf (2002), making use of the Browninan motion ancestral node reconstruction method described by Martins and Hansen (1997). Note that pPCA does ***not*** remove phylogenetic effects from principal coordinates scores, as explained Polly *et al.* (2013).

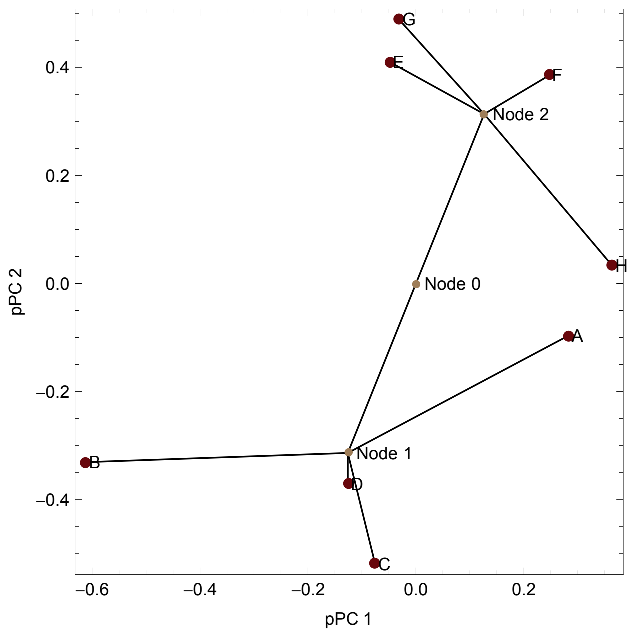
This function is a companion to the PrincipalComponentsOfShape[] and TreeToMorphospace[] functions in this package and to the ReconstructNodes[] function in the Phylogenetics package, and it requires the latter package to be installed.

Arguments:

- *proc* is a matrix of Procrustes superimposed 2D landmark coordinates with OTUs in rows and coordinates in columns.
- *labels* is a list of strings identifying each row in *proc.* These labels must be in the same order as *proc* and must match the tip names in *tree*.
- *PCs* is a list of two digits specifying which principal components will be plotted.
- *tree* is a string with the tree topology and branch lengths in Newick format, similar to that produced by the *ReadNewick* function in the *Phylogenetics for Mathematica* package.

Example:

*PhylogeneticPrincipalComponentsOfShape[proc, labels, {1,2}, tree]*



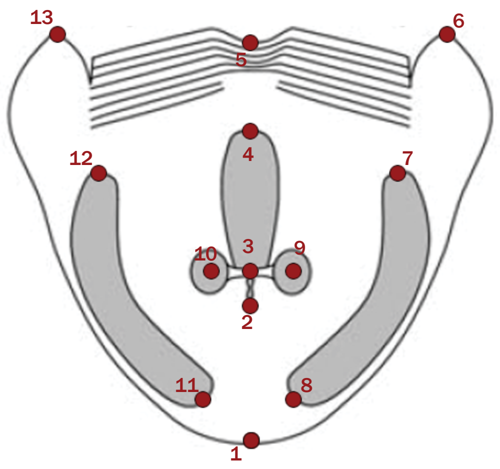
## PrincipalComponentsOfShape[*data, {PC1, PC2}, labels*]

This function does a Principal Components analysis on 2D Procrustes superimposed data (Dryden and Mardia, 1998). This analysis is the same as a Relative Warps analysis in which the alpha-weights of the Partial Warps have been set to 1 (Rohlf, 1993). The function calculates the principal components (eigenvectors) of the covariance matrix of the Procrustes residuals and displays a principal components plots of the objects, reporting the proportion of variance explained by those components. The function also displays the mean shape with each landmark numbered in order and a convex hull around the landmark configuration. Finally the function displays the principal components graph as a morphospace in which nine thin-plate spline grids have been placed to show how shape varies in the space. The grids are placed at the extreme ranges of the objects and at the mean.

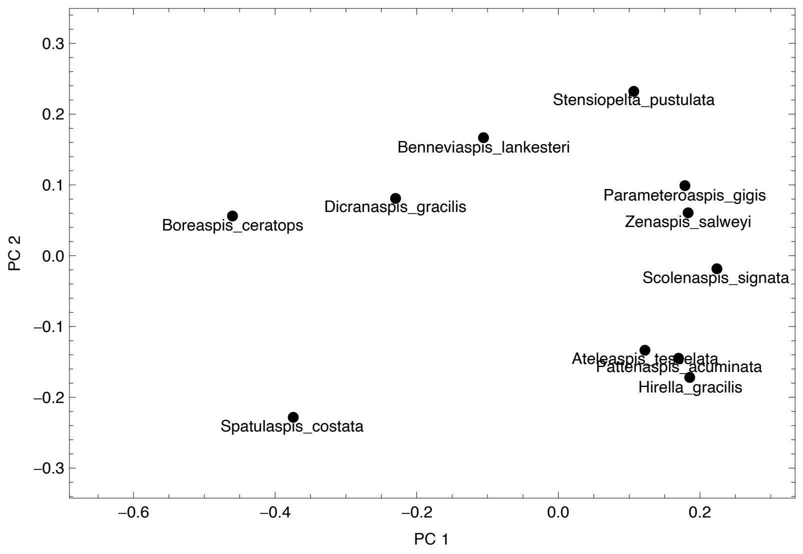
Arguments:

- *data* is a rectangular matrix of 2D Procrustes aligned landmark coordinates.
- *{PC1, PC2}* is a list of two integers specifying which principal components will be shown on the x and y axes respectively.
- *labels* is a list of strings to be used to label the objects in the principal components plot.

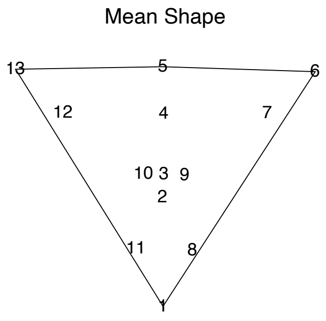
Example: Analysis of the head shields of eleven osteostracan fish species (from Sansom, 2009) using thirteen landmarks from the following scheme that have been Procrustes superimposed and stored in the variable *data*. A list of the eleven species names has been stored in *labels*.

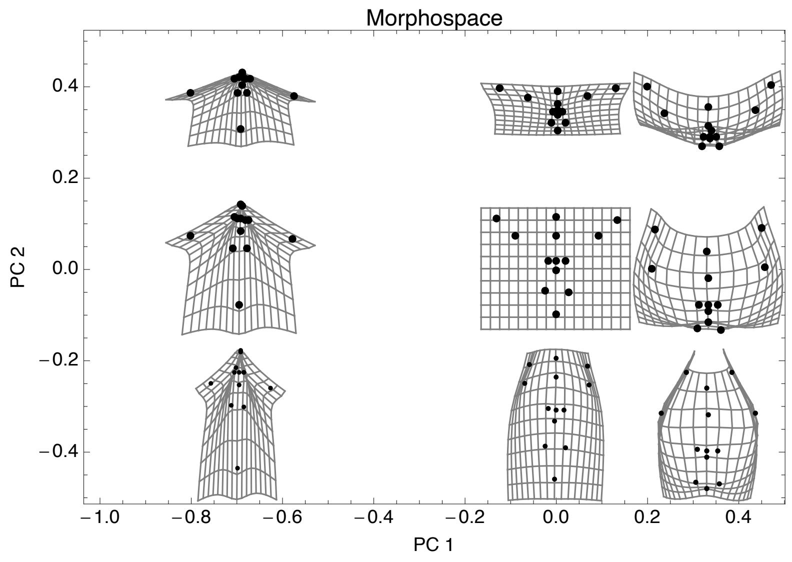


*PrincipalComponentsOfShape[data, {1,2}, labels]*



```
PC 1 explains 0.638787 of total shape variance
PC 2 explains 0.231647 of total shape variance
```





In the above example, the first plot shows the distribution of the 11 species on Principal Components 1 and 2 each labeled with the species name. The lines below the plot report the proportion of variance explained by each of the two PCs. The next plot shows the mean shape, with the numbered landmarks surrounded by a convex hull to help visualize the shape. The last plot shows the same two principal components as the first one, but this time with a series of thing plate splines to show how shape varies across the plot. The species at the left of the plot have head shields that are laterally widened at the back (landmarks 6 and 13). The species at the right have head shields that are laterally compressed at the same landmarks. The species at the top of the plot are antero-posteriorly shortened, and the ones at the bottom are elongated.

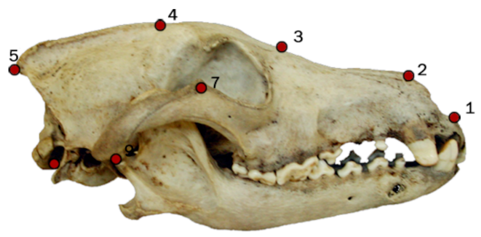
## PrincipalComponentsOfShape3D[*data, {PC1, PC2}, labels*]

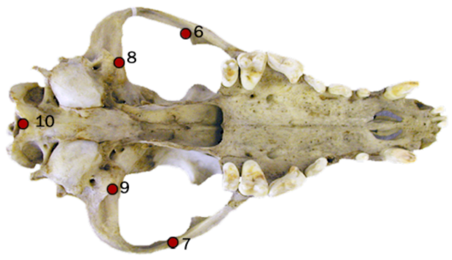
This function does a Principal Components analysis on 3D Procrustes superimposed data (Dryden and Mardia, 1998). This analysis is the same as a Relative Warps analysis in which the alpha-weights of the Partial Warps have been set to 1 (Rohlf, 1993). The function calculates the principal components (eigenvectors) of the covariance matrix of the Procrustes residuals and displays a principal components plots of the objects, reporting the proportion of variance explained by those components. The function also displays the mean shape with each landmark numbered in order with a dotted line connecting them to the shape centroid. Finally the function displays two shape models, one for each of the two PCs. Each shape model shows the mean shape with each landmark numbered in order and a vector running through it to show the variation in that landmark from one end of the PC axis to the other. The tails of the vectors indicate the positions of the landmarks at the negative end of the PC and the heads indicate the positions of the landmarks at the positive end of the PC.

Arguments:

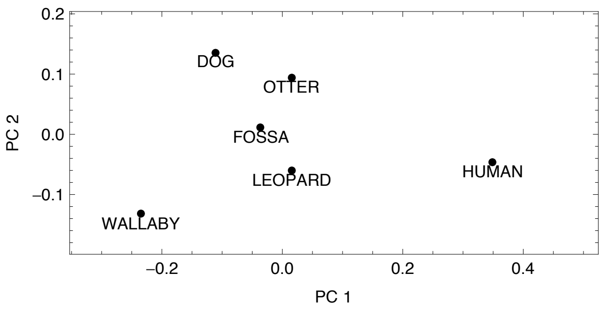
- *data* is a rectangular matrix of 3D Procrustes aligned landmark coordinates.
- *{PC1, PC2}* is a list of two integers specifying which principal components will be shown on the x and y axes respectively.
- *labels* is a list of strings to be used to label the objects in the principal components plot.

Example: Analysis of skulls of six mammals, whose names have been stored in *labels* as follows: wallaby, human, leopard, fossa, dog, and otter. Ten 3D landmarks have been placed on the cranium as follows.

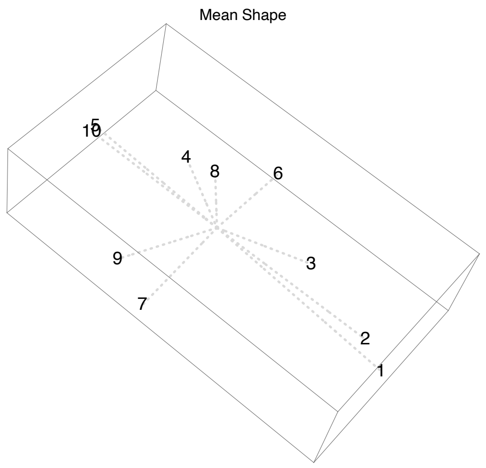




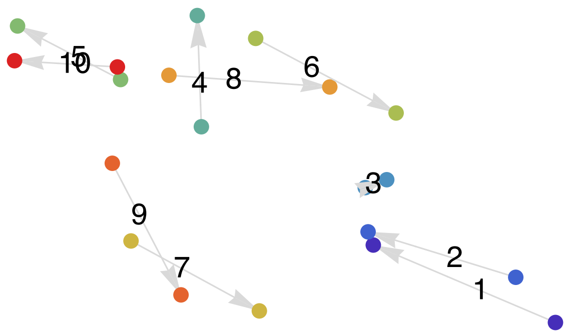
*PrincipalComponentsOfShape3D[data, {1,2}, labels]*



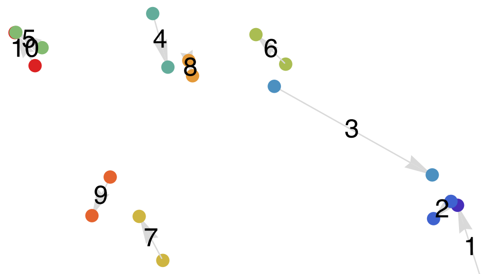
```
PC 1 explains 0.671915 of total shape variance
PC 2 explains 0.178995 of total shape variance
```



PC 1 Shape Model



PC 2 Shape Model



## ProbabilitiesOfShapesAsAncestors[*proc, labels, tree (,PCs)*]

This function calculates the probability that a shape, usually a fossil, falls within the expected distribution of an ancestral node under the assumption of a Brownian motion model of evolution. The node values and their covariances are calculated from the tree and the taxa whose names appear in its tip labels using the *ReconstructAncestors[]* function from the Phylogenetics package. P - values are then calculated for the remaining taxa in the data set. P - values are multivariate, based on all dimensions of the morphospace, unless the optional value PCs is smaller than the total number of morphospace dimensions. For each fossil, the squared Mahalanobis distance between it and the node is calculated using the scores of the fossil, the scores of the node, and the covariance matrix of the node (a diagonal matrix of the squared standard deviations associated with the evolutionary process). P is taken from a Chi -Square distribution whose degrees of freedom equal the number of morphospace dimensions being considered. The P - value is interpreted as the probability that the fossil could represent the ancestral population at that node given a Brownian motion process of evolution, the rate of evolution estimated from the tree, the topology of the tree, and the shapes of the tip taxa. Remember that the ancestral estimates are population means rather than individuals, so the interpretation is based on the fossil being representative of the mean of the population from which it came. Ideally, each tip and fossil taxon will be represented in this analysis by a sample mean shape (consensus shape) of a larger sample. P - values greater than 0.05 are highlighted in the results as being significantly compatible with the distribution of a node. Remember that a fossil that did not live at the time of the ancestor cannot logically be an ancestor, regardless of how similar its morphology is to the expected ancestral shape.  The example data set is courtesy of Aida Gómez Robles (2012).

This function is a companion to *the TreeAndFossilsToMorphospace[]* function in this package and to the *ReconstructNodes[]* function in the Phylogenetics package. This function requires the *Phylogenetics for Mathematica* package to be installed (http://mypage.iu.edu/~pdpolly/Software.html).

Arguments:

- *proc* is a matrix of Procrustes superimposed 2D landmark coordinates with OTUs in rows and coordinates in columns.
- *labels* is a list of strings identifying each row in *proc.*  These labels must be in the same order as *proc* and must match the tip names in *tree*.
- *tree* is a string with the tree topology and branch lengths in Newick format, similar to that produced by the *ReadNewick* function in the Phylogenetics for Mathematica package.
- *PCs* is an optional integer specifying the number of morphospace dimensions (PCs) to use in calculating the p-values.  By default all dimensions are used.

Example: *ProbabilitiesOfShapesAsAncestors[proc, labels, tree]*

| | Node 0 | Node 1 | Node 2 | Node 3 | Node 4 | Node 5 |
|---|---|---|---|---|---|---|
| A.africanus | **0.8920** | **0.7500** | 0.0249 | 0.0130 | 0.0002 | 0.0000 |
| H.habilis | **0.6460** | **0.9110** | **0.2520** | **0.6490** | **0.2460** | 0.0009 |
| H.ergaster | 0.0392 | 0.0022 | 0.0000 | 0.0002 | 0.0001 | 0.0000 |
| H.georgicus | **0.3120** | 0.0099 | 0.0000 | 0.0001 | 0.0000 | 0.0000 |
| H.antecessor | 0.0109 | 0.0008 | 0.0000 | 0.0001 | 0.0000 | 0.0000 |
| H.heidelbergensis | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

## ReconstructAncestorShapes[*proc, labels, tree*]

This function creates thin - plate spline representations of the shapes of taxa at the tips and nodes of a phylognetic tree. The node shapes are reconstructed using the PGLS method assuming a Brownian motion mode of evolution (see ReconstructNodes[] function in Phylogenetics package). The method essentially follows Rohlf (2002). The thin - plate spline representations are shown as the deformation of each tip and node from the shape at the root of the tree, thus showing derived changes. Note that the shape reconstructions themselves are done from a shape space centered at the mean (consensus) of the tip taxa as recommended by Rohlf (1998). The example data set is courtesy of Aida Gómez Robles (2012).

This function requires the *Phylogenetics for Mathematica* package to be installed. This function is a companion to the *TreeToMorphospace[]* function in this package and to the *ReconstructNodes[]* function in the Phylogenetics package.

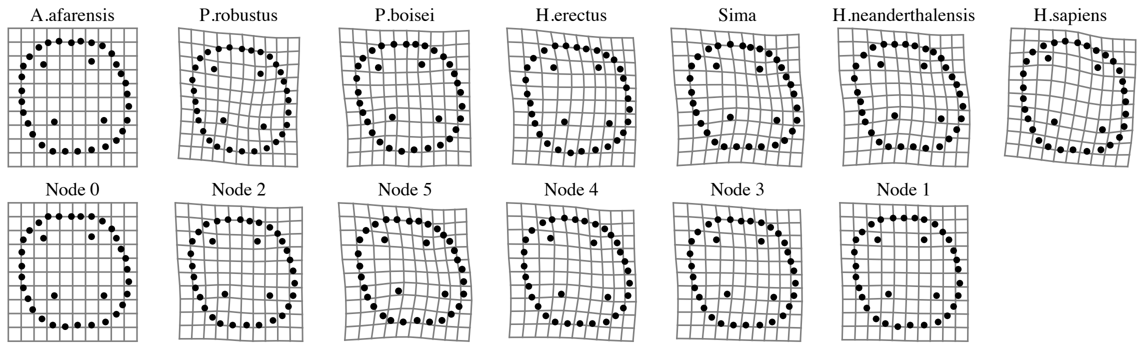
**Note:** to return the coordinates of the ancestor shapes instead of thin-plate spline images, use the function *AncestorShapes[proc, labels, tree]*, which uses the same arguments. This function returns only the shapes at the nodes and returns them in the same order as *ReconstructAncestorShapes[]* does.

Arguments:

- *proc* is a matrix of Procrustes superimposed 2D landmark coordinates with OTUs in rows and coordinates in columns.
- *labels* is a list of strings identifying each row in *proc.* These labels must be in the same order as *proc* and must match the tip names in *tree*.
- *tree* is a string with the tree topology and branch lengths in Newick format, similar to that produced by the *ReadNewick* function in the *Phylogenetics for Mathematica* package.

Example:

```
ReconstructAncestorShapes[proc , labels, tree]
```



## ShapeMANOVA[*proc, labels (, pc, iterations)*]

This function performs a one-way multivariate analysis of variance (MANOVA) of Procrustes superimposed shapes onto a grouping factor. Note that it cannot be used if one or more of the groups has fewer than three individuals. The function calculates the PC scores of the Procrustes superimposed shape, then partitions sums of squares into within and between components. All principal components (i.e., all of the shape variation) is used in the MANOVA and the results take into account the reduced degrees of freedom that result from Procrustes superimposition. Significance is assessed by non-parametric randomization, in which the real between-group sums of squares is compared to a distribution of between-group sums of squares calculated by randomizing shapes among the grouping variables. By default, 10,000 randomizations are used to produce this distribution, but the user can optionally change this with the third input parameter. The function also calculates an F-ratio and uses the F-distribution to report a parametric p-value. Note however, that the parametric p-value depends on the assumption of normality, equal variance, and equal sampling in the groups, which is seldom the case for geometric morphometric shape data. *R-squared* reports the proportion of the total variance explained by the grouping factor. The function also performs post-hoc pairwise tests between all groups with significant assessed by randomization. The same number of randomizations is used for the post-hoc tests as for the primary MANOVA. Note that one often chooses an adjusted level of significance (α) when making multiple ad-hoc comparisons, frequently the Bonferroni correction (α/*n* where *n* is the number of tests). A graphic showing the scores for each group on one PC (by default this is PC1) along with their mean and standard deviation is provided as part of the output.

Arguments:

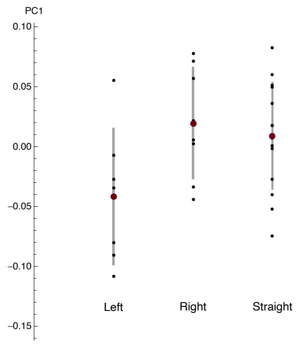
- *proc* is a matrix of Procrustes superimposed landmark coordinates with objects in rows and coordinates in columns.
- *labels* is a list of group labels.
- *pc* indicates which principal component should be shown in the graphic image. Note that all PCs are used in the tests and this option does not affect the statistical result.
- *Iterations* is an integer specifying the number of randomizations to use in calculating the significance of the group differences (by default this is 10,000).

Example:

*ShapeMANOVA[proc, labels]*

```
                                          MANOVA results         None
        | SS        df     MS             P (non-parametric)     0.0002
Between | 0.0060    2      0.0030         F-ratio                0.0330
Within  | 0.1820    27     0.0911         P (parametric)         0.0325
Total   | 0.1880    29     0.0065         R-squared              0.0320
```

```
Post-hoc tests    P (non-parametric)
Left-Right        0.0001
Left-Straight     0.0374
Right-Straight    0.0287
```



## ShapeRegress[*proc, variable (, PCs)*]

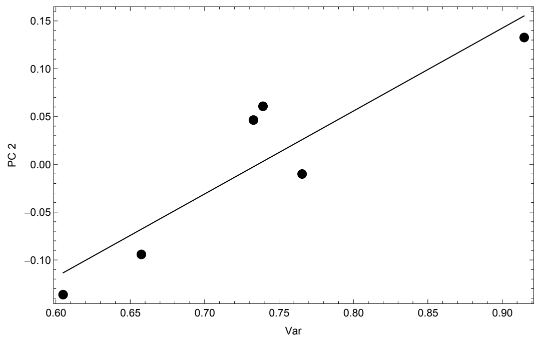
This function does a multivariate least squares regression of shape onto a single predictor variable. The function calculates the PC scores of the Procrustes superimposed shape, then regresses them onto the input variable. The function calculates an r-squared value for the regression and assesses its significance by randomization. The scores are randomized with respect to the input variable 10,000 times and the observed r-squared value compared to the randomized distribution. The function also reports the intercept, regression, and univariate r-square for each of the non-zero PCs. A graph showing the scores and regression of one PC is returned. The first PC is graphed unless a different one is specified.

Arguments:

- *proc* is a matrix of Procrustes superimposed landmark coordinates with objects in rows and coordinates in columns.
- *variable* is the variable onto which *proc* is to be regressed. It is a vector containing observations for each object from a continuous variable.
- *PCs* is an optional parameter specifying which PC should be shown in the graph. By default the regression on PC1 is shown.

Example:

*ShapeRegress[proc, x, 2]*



```
R-square (all PCs) = 0.21
P[R-square is random] = 0.34
                    | PC1        PC2       PC3         PC4       PC5
Intercept           | -0.405353  -0.63799  -0.0227621  -0.07787  0.0231763
Slope               | 0.550962   0.867165  0.0309385   0.105842  -0.0315015
Univariate R-square | 0.09       0.83      0.00        0.07      0.01
```

## TreeToMorphospace[*proc, labels, PCs, tree*]

This function projects a phylogenetic tree into a GMM morphospace using PGLS to estimate the most likely ancestral shapes under a Brownian motion model of evolution, then projecting the node shapes into the principal components space of the shape data. The method for estimating the ancestral shapes is the phylogenetic generalized least squares method (Martins and Hansen, 1997; Rohlf, 2001). The method used here essentially follows Rohlf (2002).

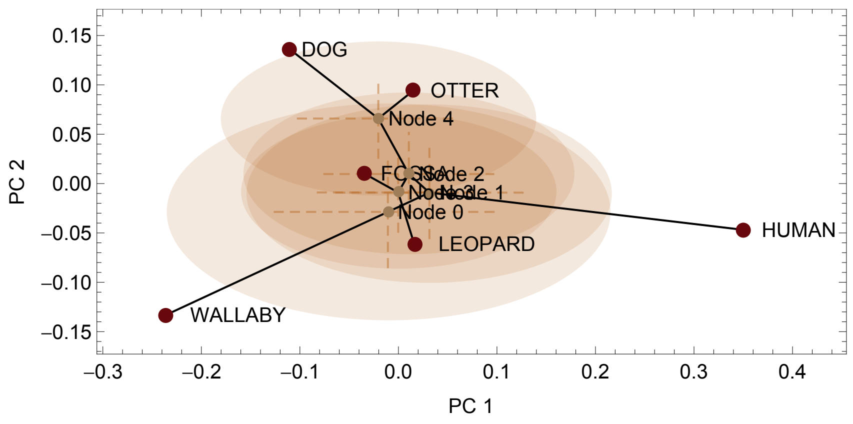
This function requires the *Phylogenetics for Mathematica* package to be installed (http://mypage.iu.edu/~pdpolly/Software.html). This function is a companion to the *ReconstructNodes[]* function in the Phylogenetics package.

Arguments:

- *proc* is a matrix of Procrustes superimposed 2D landmark coordinates with OTUs in rows and coordinates in columns.
- *labels* is a list of strings identifying each row in *proc.* These labels must be in the same order as *proc* and must match the tip names in *tree*.
- *PCs* is a list of two digits specifying which principal components will be plotted.
- *tree* is a string with the tree topology and branch lengths in Newick format, similar to that produced by the *ReadNewick* function in the *Phylogenetics for Mathematica* package.

Example:

*TreeToMorphospace[proc, labels, {1,2}, tree]*



## TreeAndFossilsToMorphospace[*proc, labels, PCs, tree*]

This function projects a phylogenetic tree and candidate ancestor shapes into a GMM morphospace. The function uses PGLS on the shapes belonging to the tree tips to estimate the most likely ancestral shapes under a Brownian motion model of evolution. It then constructs a morphospace based on the tip shapes and projects the node shapes into it and draws the tree branches. The remaining non-tip shapes, which are presumed to be fossil candidates for the ancestors, are projected into the shape space. The method for estimating the ancestral shapes is the phylogenetic generalized least squares method (Martins and Hansen, 1997; Rohlf, 2001). _The example data set is courtesy of Aida Gómez Robles (2012).

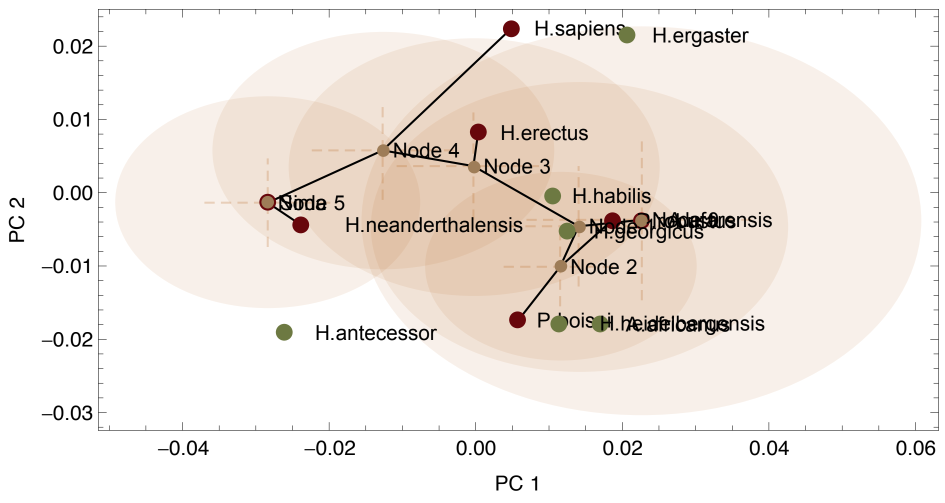
This function requires the *Phylogenetics for Mathematica* package to be installed (http://mypage.iu.edu/~pdpolly/Software.html). This function is a companion to the *ProbabilitiesofShapesAsAncestors[]* function in this package and the *ReconstructNodes[]* function in the Phylogenetics package.

Arguments:

- *proc* is a matrix of Procrustes superimposed 2D landmark coordinates with OTUs in rows and coordinates in columns.
- *labels* is a list of strings identifying each row in *proc.* These labels must be in the same order as *proc* and must match the tip names in *tree*.
- *PCs* is a list of two digits specifying which principal components will be plotted.
- *tree* is a string with the tree topology and branch lengths in Newick format, similar to that produced by the *ReadNewick* function in the Phylogenetics for Mathematica package.

Example:

*TreeAndFossilsToMorphospace[proc, labels, {1,2}, tree]*



## TwoBlockPartialLeastSquares[*data1, data2, {type1, type2}, (, PLS)*]

This function performs a two-block partial least squares analysis following the methodology of Rohlf and Corti (2000). Two blocks of data are given to the function, along with a list of two strings indicating the type of data. Allowable types are "Shape" (Procrustes superimposed coordinates), "Standardized" (independent variables with different units of measurement that need to be standardized), and "Unstandardized" (independent variables with the same unit of measurement that do not need to be  standardized).  An optional argument is the number of the PLS axis to plot in the output graph.  By default PLS 1 is plotted.

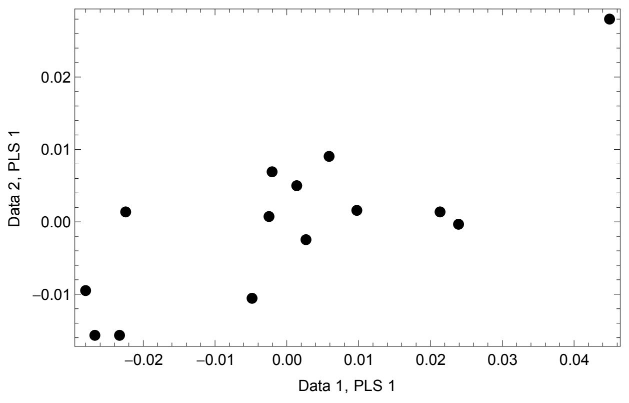
Arguments:

- *data1 and data2* are two blocks of variables, one or both of which can be a matrix of Procrustes superimposed  landmark coordinates with objects in rows and coordinates in columns.
- *{type1, type2}* is a list of two data types, in quotation marks.  Allowable types are "Shape" (Procrustes superimposed coordinates), "Standardized" (independent variables  with different units of measurement that need to be standardized), and "Unstandardized" (independent variables with the same unit of measurement that do not need to be  standardized).
- *PLS* is an integer indicating which PLS axis to plot.

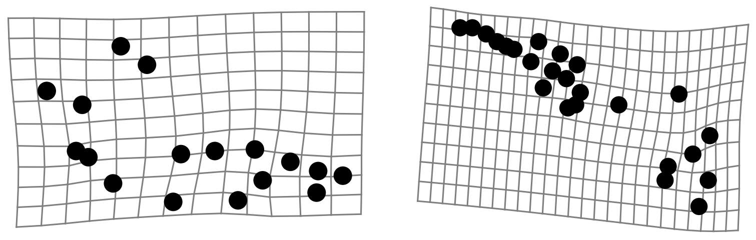
Example:

*TwoBlockPartialLeastSquares[proc1, proc2, {"Shape", "Shape"}]*

```
Proportion of Actual Squared Covariance Explained by PLS 1:  0.48
Proportion of Total Possible Squared Covariance Explained by All PLS Axes:  0.08
```



Positive End of PLS 1



## Acknowledgements

Emília Martins and F. James Rohlf provided helpful information about equations in their papers that are used in the *TreeToMorphospace[]* function and Rohlf provided documentation about TPS file formats for use in recent versions of *tpsImport[]*. Allison Bormet, Jesualdo Fuentes Gonzales, David Grossnickle, Aida Gómez Robles, Anjali Goswami, Jonathan Krieger, Michelle Lawing, and students in G562 have provided helpful feedback on the functions in this package. Aida Gómez Robles provided data used in several of the examples.

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

Adams, D. C. 2016. Evaluating modularity in morphometric data: challenges with the RV coefficient and a new test measure. *Methods in Ecology and Evolution*, **7:** 565-572.

Bookstein, F. L. 1991. *Morphometric Tools for Landmark Data: Geometry and Biology*. Cambridge University Press, Cambridge.

Dryden, I.L. and K.V. Mardia. 1998. *Statistical Shape Analysis*. John Wiley, Chichester.

Escoufier, Y. 1973. Le traitement des variables vectorielles. *Biometrics*, **29:** 751-760.

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

Hammer, O and D.A.T. Harper. 2006. *Palaeontological Data Analysis*. Blackwell Scientific, Oxford.

Klingenberg, C. P. and G. S. McIntyre. 1998. Geometric morphometrics of developmental instability: analysing patterns of fluctuating asymmetry with Procrustes methods. *Evolution*, 52: 1363-1375.

Klingenberg, C. P. 2009. Morphometric integration and modularity in configurations of landmarks: tools for evaluating a priori hypotheses. *Evolution & Development*, **11:** 405-421.

Lele, S. and J. Richtsmeier. 1995. Estimating confidence intervals for the comparison of forms. *American Journal of Physical Anthropology*, 98: 73-86.

Lele, S. and J. Richtsmeier. 2001. *An Invariant Approach to the Statistical Analysis of Shapes*. CRC Press, Boca Raton, Florida.

MacLeod, N.  1999.  Generalizing and extending the Eigenshape method of shape space visualization and analysis.  *Paleobiology*, 25: 107-138.

Manly, B.F. 1986. Randomization and regression methods for testing for associations with geographical, environmental and biological distances between populations. *Researches on Population Ecology*, 28: 201-218.

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.

Polly, P. D., A. M. Lawing, A.-C. Fabre, and A. Goswami. 2013. Phylogenetic principal components analysis and geometric morphometrics. *Hystrix*, **24:** 1-9.

Revell, L. J.  2009.  Size-correction and principal components for interspecific comparative studies.  *Evolution*, 63: 3258-3268.

Rohlf, F. J. 1993.  Relative warp analysis and an example of its application to mosquito wings.  Pp. 131-159 *in* L.F. Marcus, E. Bello, and A. Garcia-Valdecasas (eds), *Contributions to Morphometrics*.  Museo Nacional de Ciencias Naturales, Madrid, Spain.

Rohlf, F. J.  1998.  On applications of geometric morphometrics to studies of ontogeny and phylogeny.  *Systematic Biology*, 47: 147-158.

Rohlf, F. J.  1999.  Shape statistics: Procrustes superimpositions and tangent spaces. *Journal of Classification*, 16:197-223.

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

Rohlf, F. J.  2002.  Geometric morphometrics and phylogeny.  Pp. 173-193 *in* N. MacLeod and P. Forey (eds.), *Morphology, Shape and Phylogeny*.  Taylor and Francis, London.

Rohlf, F. J.  and D. Slice.  1990.  Extensions of the Procrustes method for the optimal superimposition of landmarks.  *Systematic Biology*, 39: 40-50.

Sansom, R.S.  2009.  Phylogeny, classification and characters polarity of the Osteostraci (Vertebrata).  *Journal of Systematic Palaeontology*, 7: 95-115.