Package 'geomorph'

October 13, 2021

Title Geometric Morphometric Analyses of 2D/3D Landmark Data

Version 4.0.1

Description Read, manipulate, and digitize landmark data, generate shape variables via Procrustes analysis for points, curves and surfaces, perform shape analyses, and provide graphical depictions of shapes and patterns of shape variation.

License GPL (>= 2)

URL https://github.com/geomorphR/geomorph

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends RRPP(>= 1.0.0), rgl, R(>= 3.5.0), Matrix

Imports graphics, grDevices, stats, utils, jpeg, ape, parallel, ggplot2

Suggests knitr, rmarkdown, testthat, dplyr

VignetteBuilder knitr

NeedsCompilation no

Author Dean Adams [aut, cre],
Michael Collyer [aut],
Antigoni Kaliontzopoulou [aut],
Erica Baken [aut]

Maintainer Dean Adams <dcadams@iastate.edu>

Repository CRAN

Date/Publication 2021-10-13 16:32:04 UTC

R topics documented:

`geomorph-package` *Geometric morphometric analyses for 2D/3D data*

Description

Functions in this package allow one to read, manipulate, and digitize landmark data; generate shape variables via Procrustes analysis for points, curves and surface data, perform statistical analyses of shape variation and covariation, and provide graphical depictions of shapes and patterns of shape variation.

geomorph TOC

geomorph-package

Author(s)

Dean C. Adams, Michael Collyer, Antigoni Kaliontzopoulou, and Erica Baken

---

`arrayspecs` *Convert landmark data matrix into array (p x k x n)*

Description

Convert a matrix of landmark coordinates into a three-dimensional array

Usage

```
arrayspecs(A, p, k, sep = NULL)
```

Arguments

A	A matrix containing landmark coordinates for a set of specimens
p	Number of landmarks
k	Number of dimensions (2 or 3)
sep	An optional argument to attempt to separate variable names into landmark dimension and landmark number variables. For example, X.1, Y.1, Z.1, X.2, Y.2, Z.2, ..., can be separated with sep = ".", such that rows of landmark configurations are labeled 1, 2, 3, ..., and columns are labeled X, Y, Z. Note, for variables, X1, Y1, Z1, X2, Y2, Z2, ..., where no separator is evident, use sep = "". Any illogical separation argument will result in unlabeled variables. If sep = NULL (the default), unlabeled variables are forced. This is a good idea if the original matrix has landmarks out of order (as the the landmark labels might not sort as expected).

Details

This function converts a matrix of landmark coordinates into a 3D array (p x k x n), which is the required input format for many functions in geomorph. The input matrix can be arranged such that the coordinates of each landmark are found on a separate row, or that each row contains all landmark coordinates for a single specimen.

Value

Function returns a 3D array (p x k x n), where p is the number of landmark points, k is the number of landmark dimensions (2 or 3), and n is the number of specimens. The third dimension of this array contains names for each specimen if specified in the original input matrix.

Author(s)

Dean Adams & Mike Collyer

See Also

two.d.array

Examples

```
x<-matrix(rnorm(18),nrow=3)  # Random triangles (all coordinates on same
 # row for each triangle)
arrayspecs(x,3,2)

x2<-matrix(rnorm(18),ncol=2) # Random triangles (each landmark on its
# own row)
arrayspecs(x2,3,2)
```

`bilat.symmetry` *Analysis of bilateral symmetry*

Description

Function performs an analysis of directional and fluctuating asymmetry for bilaterally symmetric objects

Usage

```
bilat.symmetry(
  A,
  ind = NULL,
  side = NULL,
  replicate = NULL,
  object.sym = FALSE,
  land.pairs = NULL,
  data = NULL,
  iter = 999,
  seed = NULL,
  RRPP = TRUE,
  SS.type = c("I", "II", "III"),
  turbo = TRUE,
  Parallel = FALSE,
  print.progress = TRUE,
  ...
)
```

Arguments

`A`	One of either A 3D array (p x k x n) containing raw landmarks (requiring GPA to be performed) or a gpagen object (if GPA has been previously performed) or a geomorphShapes object (requiring GPA to be performed). Any gpagen argument should work within bilat.symmetry.
`ind`	A vector containing labels for each individual. For matching symmetry, the matched pairs receive the same label (replicates also receive the same label).
`side`	An optional vector (for matching symmetry) designating which object belongs to which 'side-group'
`replicate`	An optional vector designating which objects belong to which group of replicates. Alternatively, this can be a character value to indicate the name of the variable in the data frame to use.
`object.sym`	A logical value specifying whether the analysis should proceed based on object symmetry =TRUE or matching symmetry =FALSE
`land.pairs`	An optional matrix (for object symmetry) containing numbers for matched pairs of landmarks across the line of symmetry

data	A data frame for the function environment, see geomorph.data.frame. It is imperative that the variables "ind", "side", and "replicate" in the data frame match these names exactly (as shown in examples below).
iter	Number of iterations for significance testing.
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
RRPP	A logical value indicating whether residual randomization should be used for significance testing.
SS.type	A choice between type I (sequential), type II (hierarchical), or type III (marginal).
turbo	A logical value that if TRUE, suppresses coefficient estimation in every random permutation, in order to speed up computation time.
Parallel	Either a logical value to indicate whether parallel processing should be used or a numeric value to indicate the number of cores to use in parallel processing via the parallel library. If TRUE, this argument invokes forking of all processor cores, except one. If FALSE, only one core is used. A numeric value directs the number of cores to use, but one core will always be spared.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.
...	Arguments to pass onto gpagen

Details

The function quantifies components of shape variation for a set of specimens as described by their patterns of symmetry and asymmetry. Here, shape variation is decomposed into variation among individuals, variation among sides (directional asymmetry), and variation due to an individual x side interaction (fluctuating symmetry). These components are then statistically evaluated using Procrustes ANOVA. Statistical assessment of model effects for shape variation is accomplished using permutation procedures. Methods for both matching symmetry and object symmetry can be implemented. Matching symmetry is when each object contains mirrored pairs of structures (e.g., right and left hands) while object symmetry is when a single object is symmetric about a midline (e.g., right and left sides of human faces). Details on general approaches for the study of symmetry in geometric morphometrics may be found in: Mardia et al. 2000; Klingenberg et al. 2002.

As input, the function receives either A 3D array (p x k x n) containing raw landmarks (requiring GPA to be performed) or a gpagen object (if GPA has been previously performed) or a geomorphShapes object. If one wishes to incorporate semilandmarks, GPA can either be performed first using gpagen, or within bilat.symmetry by passing adequate GPA arguments (i.e. curves, surfaces, ProcD etc, see gpagen. If a geomorphShapes object is provided, semilandmarks are automatically identified and slid during GPA. For "object.sym = FALSE, landmarks should be of dimension (p x k x 2n), as each specimen is represented by both left and right configurations.

Analyses of symmetry for matched pairs of objects is implemented when object.sym=FALSE. Here, a 3D array [p x k x 2n] contains the landmark coordinates for all pairs of structures (2 structures for each of n specimens). Because the two sets of structures are on opposite sides, they represent mirror

images, and one set must be reflected prior to the analysis to allow landmark correspondence. IT IS ASSUMED THAT THE USER HAS DONE THIS PRIOR TO PERFORMING THE SYMMETRY ANALYSIS. Reflecting a set of specimens may be accomplished by multiplying one coordinate dimension by '-1' for these structures (either the x-, the y-, or the z-dimension). A vector containing information on individuals and sides must also be supplied. Replicates of each specimen may also be included in the dataset, and when specified will be used as measurement error (see Klingenberg and McIntyre 1998).

Analyses of object symmetry is implemented when object.sym=TRUE. Here, a 3D array [p x k x n] contains the landmark coordinates for all n specimens. To obtain information about asymmetry, the function generates a second set of objects by reflecting them about one of their coordinate axes. The landmarks across the line of symmetry are then relabeled to obtain landmark correspondence. The user must supply a list of landmark pairs. A vector containing information on individuals must also be supplied. Replicates of each specimen may also be included in the dataset, and when specified will be used as measurement error.

The function also provides individual measures of signed and unsigned asymmetry, calculated as the Procrustes distance between the right and left element (for paired structures, as detailed in Klingenberg and McIntyre 1998) or side of the structure (for object symmetry, following Lazić et al 2015). The computational difference betwen the two approaches consists in that, for object symmetry, only paired landmarks are considered, excluding the landmarks of the midline.

Notes for geomorph 3.0:

Compared to older versions of geomorph, some results can be expected to be slightly different. Starting with geomorph 3.0, results use only type I sums of squares (SS) with either full randomization of raw shape values or RRPP (preferred with nested terms) for analysis of variance (ANOVA). Older versions used a combination of parametric and non-parametric results, as well as a combination of type I and type III SS. While analytical conclusions should be consistent (i.e., "significance" of effects is the same), these updates maintain consistency in analytical philosophy. This change will require longer computation time for large datasets, but the trade-off allows users to have more flexibility and eliminates combining disparate analytical philosophies.

Note also that significance of terms in the model are found by comparing F-values for each term to those obtained via permutation. F-ratios and df are not strictly necessary (a ratio of SS would suffice), but they are reported as is standard for anova tables. Additionally, users will notice that the df reported are based on the number of observations rather than a combination of objects * coordinates * dimensions, as is sometimes found in morphometric studies of symmetry. However, this change has no effect on hypothesis testing, as only SS vary among permutations (df, coordinates, and dimensions are constants).

The generic functions, `print`, `summary`, and `plot` all work with `bilat.symmetry`.

Value

An object of class "bilat.symmetry" returns a list of the following

`shape.anova`	An analysis of variance table for the shape data.
`size.anova`	An analysis of variance table for the shape data (when object.sym = FALSE).
`symm.shape`	The symmetric component of shape variation.
`asymm.shape`	The asymmetric component of shape variation.

DA.component	The directional asymmetry component, found as the mean shape for each side.
FA.component	The fluctuating asymmetry component for each specimen, found as the specimen specific side deviation adjusted for the mean directional asymmetry in the dataset.
signed.AI	Individual signed asymmetry index, as per Klingenberg and McIntyre, 1998; Lazić et al 2015.
#'	
unsigned.AI	Individual unsigned asymmetry index, as per Klingenberg and McIntyre, 1998; Lazić et al 2015.
data.type	A value indicating whether the analysis was performed as Object or Matching symmetry.
permutations	The number of random permutations used.
random.shape.F	A matrix of random F-values from the Shape analysis.
random.size.F	A matrix of random F-values from the Centroid Size analysis (when object.sym = FALSE).
perm.method	A value indicating whether "Raw" values were shuffled or "RRPP" performed.
procD.lm.shape	A list of typical output from an object of class procD.lm, for shape
procD.lm.size	If applicable, a list of typical output from an object of class procD.lm, for size (when object.sym = FALSE).
call	The matched call.

Author(s)

Dean Adams, Michael Collyer and Antigoni Kaliontzopoulou

References

Klingenberg, C.P. and G.S. McIntyre. 1998. Quantitative genetics of geometric shape in the mouse mandible. Evolution. 55:2342-2352.

Mardia, K.V., F.L. Bookstein, and I.J. Moreton. 2000. Statistical assessment of bilateral symmetry of shapes. Biometrika. 87:285-300.

Klingenberg, C.P., M. Barluenga, and A. Meyer. 2002. Shape analysis of symmetric structures: quantifying variation among individuals and asymmetry. Evolution. 56:1909-1920.

Lazić, M. M., M. A. Carretero, J. Crnobrnja-Isailović, and A. Kaliontzopoulou. 2015. Effects of environmental disturbance on phenotypic variation: an integrated assessment of canalization, developmental stability, modularity, and allometry in lizard head shape. The American Naturalist 185:44–58.

Examples

```
#Example of matching symmetry

data(mosquito)
gdf <- geomorph.data.frame(wingshape = mosquito$wingshape,
```

```
ind=mosquito$ind,
side=mosquito$side,
replicate=mosquito$replicate)
mosquito.sym <- bilat.symmetry(A = wingshape, ind = ind, side = side,
replicate = replicate, object.sym = FALSE, RRPP = TRUE, iter = 149,
data = gdf)
summary(mosquito.sym)
plot(mosquito.sym, warpgrids = TRUE)
mosquito.sym$shape.anova # extract just the anova table on shape

# Previous example, performing GPA first
Y.gpa <- gpagen(mosquito$wingshape)
mosquito.sym2 <- bilat.symmetry(A = Y.gpa, ind = ind, side = side,
replicate = replicate, object.sym = FALSE, RRPP = TRUE, iter = 149,
data = gdf)
summary(mosquito.sym2)
summary(mosquito.sym) # same results

#Example of object symmetry

data(lizards)
gdf <- geomorph.data.frame(shape = lizards$coords,
ind = lizards$ind,
replicate = lizards$rep)
liz.sym <- bilat.symmetry(A = shape, ind = ind, rep = rep,
object.sym = TRUE,
land.pairs = lizards$lm.pairs, data = gdf, RRPP = TRUE, iter = 149)
summary(liz.sym)

# Example of object symmetry in 3D and including semilandmarks

data(scallops)
gdf <- geomorph.data.frame(shape = scallops$coorddata,
ind = scallops$ind)
scallop.sym <- bilat.symmetry(A = shape, ind = ind,
object.sym = TRUE,
curves= scallops$curvslide, surfaces = scallops$surfslide,
land.pairs=scallops$land.pairs, data = gdf, RRPP = TRUE, iter = 149)
summary(scallop.sym)
# NOTE one can also: plot(scallop.sym, warpgrids = TRUE, mesh = NULL)
# NOTE one can also: scallop.sym$data.type # recall the symmetry type
```

`buildtemplate` *Build 3D surface template*

Description

An interactive function to build a template for the digitization across specimens of three dimensional (3D) surface sliding semilandmarks. Input for the function is either a matrix of vertex coordinates defining a 3D surface object or a mesh3d object as obtained from `read.ply`.

Usage

```
buildtemplate(spec, fixed, center = TRUE, surface.sliders, ptsize = 1)
```

Arguments

spec	An object of class shape3d/mesh3d, or matrix of 3D vertex coordinates
fixed	Either a numeric value designating the number of fixed landmarks to be selected by digit.fixed, or a matrix of 3D coordinates collected previously
center	Should the object 'spec' be centered prior to digitizing?
surface.sliders	The number of desired surface sliders
ptsize	Size of mesh points (vertices), e.g. 0.1 for dense meshes, 3 for sparse meshes

Details

Function constructs a template of surface slider semilandmarks. If desired, the user can simultaneously digitize the fixed points (see digitizing below), however these may have been previously digitized separately using digit.fixed.

The function finds surface semilandmarks, chosen automatically from the mesh at roughly equal distances, using the nearest-neighbor algorithm outlined in Gunz et al. (2005) and Mitteroecker and Gunz (2009).

The set of fixed and surface slider landmarks are saved in the working directory as a txt file named "template". This file will then be used to extract a set of similarly numbered surface semilandmarks on subsequent specimens using the function digitsurface. Because template matching is based on the correspondence of fixed landmark points in the template and the target specimen, a minimum of four fixed landmarks must be used. However, to ensure a strong match between the scan and the template, it is recommended that a higher number of fixed points is used.

For more details see the vignette: `vignette("geomorph.digitize3D")`.

NOTE: Function centers the mesh before digitizing by default (center=TRUE). If one chooses not to center, specimen may be difficult to manipulate in rgl window.

Digitizing: Digitizing of fixed landmarks is interactive. Once a point is selected, the user is asked if the system should keep or discard the selection (y/n). If "y", the user is asked to continue to select the next landmark. If "n" the removes the last chosen landmark, and the user is asked to select it again. This can be repeated until the user is comfortable with the landmark chosen.

To digitize with a standard 3-button (PC):

1. the RIGHT mouse button (primary) to select points to be digitized,
2. the LEFT mouse button (secondary) is used to rotate mesh,
3. the mouse SCROLLER (third/middle) is used to zoom in and out.

NOTE: Digitizing functions on MACINTOSH computers using a standard 3-button mice works as specified. Macs using platform specific single button mice, XQuartz must be configured: go to Preferences > Input > tick "Emulate three button mouse":

1. press button to rotate 3D mesh,
2. press button while pressing COMMAND key to select vertex to be used as a landmark),

3. press button while pressing OPTION key to adjust mesh perspective.
4. the mouse SCROLLER or trackpad two finger scroll is used to zoom in an out.

NOTE: there is no pan (translate) functionality in rgl library for all platforms at this time. The template can be edited using function `editTemplate`.

AUTO mode:

The function as described above (for interactive mode) calls `digit.fixed`, prompting the user to select fixed landmarks in the rgl window. However if the user has digitized these fixed landmark elsewhere (e.g., in other software), then the input for parameter 'fixed' can be a p-x-k matrix of 3D coordinates. In this case, the function will automatically use these landmarks to build the template of sliding semilandmarks.

Value

The function writes to the working directory three files: an NTS file with the name of the specimen and .nts suffix containing the landmark coordinates, "template.txt" containing the same coordinates for use with the function `digitsurface`, and "surfslide.csv", a file containing the address of the landmarks defined as "surface sliders" for use with `gpagen`. The function also returns to the console an n x 3 matrix containing the x,y,z coordinates of the digitized landmarks.

Author(s)

Erik Otarola-Castillo & Emma Sherratt

References

Gunz P, Mitteroecker P, & Bookstein FJ (2005) Semilandmarks in Three Dimensions. Modern Morphometrics in Physical Anthropology, ed Slice DE (Springer-Verlag, New York), pp 73-98.

Mitteroecker P & Gunz P (2009) Advances in Geometric Morphometrics. Evolutionary Biology 36(2):235-247.

See Also

`read.ply`

`digit.fixed`

`digitsurface`

`combine.subsets` *Combine separate landmark configurations*

Description

Combine separate landmark configurations (subsets) into one landmark set

Usage

```
combine.subsets(
  ...,
  gpa = TRUE,
  CS.sets = NULL,
  norm.CS = FALSE,
  weights = NULL
)
```

Arguments

...	Class gpagen objects, Procrustes shape variables from class gpagen objects, or original landmarks. As many data sets as desired can be supplied, separated by commas. Additionally, arguments passed onto gpagen can be provided, but these arguments will be passed onto all GPAs performed. Therefore, it is recommended that GPA is performed first with gpagen, to maintain flexibility. Naming subsets is a good idea, as landmark names in the combined data set will take the subset name as a precursor.
gpa	A logical argument to indicate if either (1) GPA should be performed (if original landmarks are provided) or (2) gpagen objects are provided. If TRUE, this function will check to see if the input is an object of class gpagen, and if not, perform GPA. If FALSE, landmarks will be unchanged. (One would choose gpa = FALSE if inputting aligned coordinates and centroid size, separately. There might be little reason to do this, unless one wishes to intentionally not scale configurations.)
CS.sets	A list, array, or matrix of centroid sizes to use for scaling. The default is NULL and should be left so if gpa = TRUE. If gpa = FALSE and CS.set is null, all centroid sizes become 1.0, meaning no scaling of configurations by relative size is performed. If gpa = FALSE and CS.set is provided, scaling by relative size is performed according to the data input (One could weight configurations via this method.). If the CS.set input is a matrix, it is assumed that rows are specimens and columns correspond to the different landmark sets. Lists or arrays should be in the same order as the landmark sets.
norm.CS	An option to normalize centroid size, according to the method of Dryden and Mardia (2016). If TRUE, centroid sizes are divided by the square root of the number of landmarks. This may have some appeal when one configuration is landmark-dense and another is landmark-sparse, but both correspond to structures of similar surface area or volume. Using this option should be done with caution, as it can make small configurations larger than large configurations, in a relative sense. Choosing this option will probably produce relative centroid sizes that are more equal, irrespective of the number of landmarks.
weights	An option to define (positive) weights used in calculation of relative centroid sizes (Collyer et al. 2020). Note that this option, if not NULL, will override norm.CS, as normalizing CS is one method of weighting centroid size. Using this option should be done with caution, as it can make small configurations larger than large configurations, in a relative sense. Note that no adjustments of weights are made - the user must define the weights exactly as intended.

Details

This function combines landmark configurations (either landmarks requiring GPA or Procrustes shape variables following GPA) to create a different morphological data set. This might be of interest, for example, if one has landmarks digitized on separate images collected from the same organisms. (In the examples below, configurations for heads and tails of larval salamanders were collected separately from images taken on the same individuals.) An attempt is made to scale configurations by their relative centroid sizes, following the procedure in Davis et al. (2016); i.e., landmark coordinates are multiplied by CSi/sqrt(CSi^2 + CSj^2 + ...) before combining them, so that resulting combinations of landmarks are scaled to unit centroid size. This is only possible if GPA is performed on landmarks (gpa = TRUE) or centroid sizes are provided as an argument. Objects of class `gpagen` can be used rather than original landmarks (recommended, especially if curves or surface sliding semilandmarks are used, as different arguments cannot be passed onto onto separate GPAs via this function).

The procedure of Davis et al. (2016) is an extension of the "separate subsets" method of Adams (1999) for articulated structures.

Value

An object of class `combined.set` is a list containing the following

`cords`	An [p x k x n] array of scaled, concatenated landmark coordinates.
`CS`	A matrix of columns representing original centroid sizes of subsets, either input or found via GPA.
`GPA`	If gpa = TRUE, the gpagen results for each subset.
`gpa.coords.by.set`	A list of the coordinates following GPA for each subset.
`adj.coords.by.set`	A list of the coordinates of each subset, after rescaling.
`points.by.set`	A vector of the number of landmarks in each subset.

Author(s)

Michael Collyer

References

Davis, M.A., M.R. Douglas, M.L. Collyer, & M.E. Douglas, M. E. 2016. Deconstructing a species-complex: geometric morphometric and molecular analyses define species in the Western Rattlesnake (Crotalus viridis). PloS one, 11(1), e0146166.

Adams, D.C. 1999. Methods for shape analysis of landmark data from articulated structures. Evolutionary Ecology Research. 1:959-970.

Dryden, I.L. and K.V Mardia. 2016. Statistical shape analysis, with applications in R: Second edition.

Collyer, M.L., M.A. Davis, and D.C. Adams. 2020. Making heads or tails of combined landmark configurations in geometric morphometric data. Evolutionary Biology. 47:193-205.

Examples

```
data(larvalMorph)
head.gpa <- gpagen(larvalMorph$headcoords,
  curves = larvalMorph$head.sliders)
tail.gpa <- gpagen(larvalMorph$tailcoords,
  curves = larvalMorph$tail.sliders)

# Combine original data without GPA (plot to see relative size of
# heads and tails)

 all.lm <- combine.subsets(head = larvalMorph$headcoords,
 tail = larvalMorph$tailcoords, gpa = FALSE, CS.sets = NULL)
 plotAllSpecimens((all.lm$coords))

 # Combine with GPA and relative centroid size

comb.lm <- combine.subsets(head = head.gpa, tail = tail.gpa, gpa = TRUE)
summary(comb.lm)

# (configurations are actual relative size)
comb.lm$coords[,,1]

# Plot all specimens and just first specimen and color code landmarks
par(mfrow = c(1,2))
plotAllSpecimens(comb.lm$coords)
plot(comb.lm$coords[,,1], pch = 21, bg = c(rep(1,26),
rep(2,64)), asp = 1)

# Override relative centroid size

comb.lm <- combine.subsets(head = head.gpa$coords,
tail = tail.gpa$coords, gpa = FALSE, CS.sets = NULL)
par(mfrow = c(1,2))
plotAllSpecimens(comb.lm$coords)
plot(comb.lm$coords[,,1], pch = 21, bg = c(rep(1,26),
rep(2,64)), asp = 1)

# Note the head is as large as the tail, which is quite unnatural.

## Normalizing centroid size

comb.lm <- combine.subsets(head = head.gpa,
tail = tail.gpa, gpa = TRUE, norm.CS = TRUE)
summary(comb.lm)
par(mfrow = c(1,2))
plotAllSpecimens(comb.lm$coords)
plot(comb.lm$coords[,,1], pch = 21, bg = c(rep(1,26),
rep(2,64)), asp = 1)
par(mfrow = c(1,1))

# Note that the head is too large, compared to a real specimen.
# This option focuses on average distance of points to centroid,
```

```
# but ignores the number of landmarks.
# Consequently,the density of landmarks in the head and tail are
# irrelevant and the head size is inflated because of the fewer
# landmarks in the configuration.

## Weighting centroid size

comb.lm <- combine.subsets(head = head.gpa,
tail = tail.gpa, gpa = TRUE, norm.CS = FALSE, weights = c(0.3, 0.7))
summary(comb.lm)
par(mfrow = c(1,2))
plotAllSpecimens(comb.lm$coords)
plot(comb.lm$coords[,,1], pch = 21, bg = c(rep(1,26),
rep(2,64)), asp = 1)
par(mfrow = c(1,1))

# Note that the head is way too small, compared to a real specimen.
# This option allows one to dictate the relative sizes of subsets
#  as portions of the combined set.  An option like this should be
# used with caution, but can help overcome issues caused by landmark
# density.
```

`compare.CR` *Comparisons of Effect Sizes from Modularity Analyses*

Description

Function performs an analysis to compare the effect sizes of two or more CR effects

Usage

```
compare.CR(..., CR.null = TRUE, two.tailed = TRUE)
```

Arguments

`...`	saved analyses of class CR
`CR.null`	A logical value to indicate whether a Null CR model (no modularity) should also be included in analysis.
`two.tailed`	A logical value to indicate whether a two-tailed test (typical and default) should be performed.

Details

The function statistically compares the effect sizes of two or more CR analyses. Typically, this function might be used to compare levels of modularity between two or more samples, each measuring the degree of morphological modularity in each. Alternatively, the approach can compare the degree of modular signal as expressed by alternative modular hypotheses for the same dataset.

The analysis calculates effect sizes as standard deviates, z, and performs two-sample z-tests, using the pooled standard error from the sampling distributions of the CR analyses. The method follows that of Adams and Collyer (2019) used to compare patterns of modularity across datasets.

To use this function, simply perform `modularity.test`, or `phylo.modularity` on as many samples or alternative modular hypotheses as desired. Any number of objects of class CR can be input. For the case of the latter, one may wish to include the null hypothesis of no modularity (i.e., that all variables belong to a single module). For this, the CR.null = TRUE option should be specified. Finally, one may perform the comparison as either a one-tailed or a two-tailed (default) test.

Value

An object of class compare.CR, returns a list of the following

`sample.z`	A vector of effect sizes for each sample.
`sample.r.sd`	A vector of standard deviations for each sampling distribution (following Box-Cox transformation).
`pairwise.z`	A matrix of pairwise, two-sample z scores between all pairs of effect sizes.
`pairwise.p`	A matrix of corresponding P-values.

Author(s)

Dean Adams and Michael Collyer

References

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes for morphometric data. *Evolution*. 73:2352-2367.

Examples

```
#NOT RUN
# Example 1: Compare modular signal across datasets

# data(pupfish)
# Y.gpa<-gpagen(pupfish$coords, print.progress = FALSE)    #GPA-alignment

## landmarks on the body and operculum
# land.gps<-rep('a',56); land.gps[39:48]<-'b'

# group <- factor(paste(pupfish$Pop, pupfish$Sex, sep = "."))
# levels(group)

# coords.gp <- coords.subset(Y.gpa$coords, group)

# modul.tests <- Map(function(x) modularity.test(x, land.gps,iter=999,
# print.progress = FALSE), coords.gp)

# the map function performs the integration test on each 3D array
```

```
# in the lists provided

#  modul.tests$Marsh.F
#  modul.tests$Marsh.M
#  modul.tests$Sinkhole.F
#  modul.tests$Sinkhole.M

# group.Z <- compare.CR(modul.tests, CR.null = FALSE)
# summary(group.Z)

# Example 2: Compare alternative modular hypotheses

# 3 module hypothesis (tail now a module)
# land.gps3 <- rep('a',56); land.gps3[39:48]<-'b';
# land.gps3[c(6:9,28:38)] <- 'c'

# 4 module hypothesis (eye now a module)
# land.gps4 <- rep('a',56); land.gps4[39:48]<-'b';
# land.gps4[c(6:9,28:38)] <- 'c';
#  land.gps4[c(10,49:56)] <- 'd'

# m3.test <- modularity.test(coords.gp$Marsh.F,land.gps3, iter = 499,
# print.progress = FALSE)
# m4.test <- modularity.test(coords.gp$Marsh.F,land.gps4, iter = 499,
# print.progress = FALSE)

# model.Z <- compare.CR(modul.tests$Marsh.F,m3.test,m4.test,
# CR.null = TRUE)
# summary(model.Z)
```

`compare.evol.rates` *Comparing net rates of shape evolution on phylogenies*

Description

Function calculates net rates of shape evolution for two or more groups of species on a phylogeny from a set of Procrustes-aligned specimens

Usage

```
compare.evol.rates(
  A,
  phy,
  gp,
  iter = 999,
  seed = NULL,
  method = c("permutation", "simulation"),
  print.progress = TRUE
)
```

Arguments

A	A 3D array (p x k x n) containing GPA-aligned coordinates for all specimens, or a matrix (n x variables)
phy	A phylogenetic tree of class phylo - see read.tree in library ape
gp	A factor array designating group membership for individuals
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
method	One of "simulation" or "permutation", to choose which approach should be used to assess significance.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function compares net rates of morphological evolution for two or more groups of species on a phylogeny, under a Brownian motion model of evolution. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. The approach is based on the outer-product matrix of between species differences in morphospace after phylogenetic transformation (Adams 2014). From the data the net rate of shape evolution for each group in the multi-dimensional space is calculated, and a ratio of rates is obtained. If three or more groups of species are used, the ratio of the maximum to minimum rate is used as a test statistic (see Adams 2014). The function can be used with univariate data (i.e. centroid size) if imported as matrix with rownames giving the taxa names.

The generic functions, print, summary, and plot all work with compare.evol.rates. The generic function, plot, produces a histogram of random rate-ratios associated with the resampling procedure.

Notes for geomorph 3.0.4 and subsequent versions:

Significance testing is now accomplished in one of two ways. First, phylogenetic simulation may be used, in which tips data are obtained under Brownian motion using a common evolutionary rate pattern for all species on the phylogeny. Specifically, the common evolutionary rate matrix for all species is used, with the multi-dimensional rate used along the diagonal elements (see Denton and Adams 2015). This procedure is more general than the original simulation procedure, and retains the desirable statistical properties of earlier methods, and under a wider array of data types. Second, significance may be accomplished via permutation, where data values at the tips are permuted relative to the (see Adams and Collyer 2018). This procedure is shown to retain all appropriate statistical properties, including rotation-invariance of significance levels (see results of Adams and Collyer 2018). In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019). Values from these distributions are log-transformed prior to effect size estimation, to assure normally distributed data.

Value

An object of class "evolrate" returns a list with the following components:

`sigma.d.ratio`	The ratio of maximum to minimum net evolutionary rates.
`P.value`	The significance level of the observed ratio.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`sigma.d.gp`	The phylogenetic net evolutionary rate for each group of species on the phylogeny.
`random.sigma`	The sigma values found in random permutations of the resampling procedure.
`permutations`	The number of random permutations used.

Author(s)

Dean Adams & Emma Sherratt

References

Adams, D.C. 2014. Quantifying and comparing phylogenetic evolutionary rates for shape and other high-dimensional phenotypic data. Syst. Biol. 63:166-177.

Denton, J.S.S., and D.C. Adams. 2015. A new phylogenetic test for comparing multiple high-dimensional evolutionary rates suggests interplay of evolutionary rates and modularity in lanternfishes (Myctophiformes; Myctophidae). Evolution. 69:2425-2440.

Adams, D.C. and M.L. Collyer. 2018. Multivariate comparative methods: evaluations, comparisons, and recommendations. Systematic Biology. 67:14-31.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

Examples

```
data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment
 gp.end<-factor(c(0,0,1,0,0,1,1,0,0))  #endangered species vs. rest
 names(gp.end)<-plethspecies$phy$tip

ER<-compare.evol.rates(A=Y.gpa$coords, phy=plethspecies$phy,
  method="simulation",gp=gp.end,iter=999)
summary(ER)
plot(ER)
```

compare.multi.evol.rates *Comparing net rates of evolution among traits on phylogenies*

Description

Function calculates net rates of shape evolution for two or more multi-dimensional traits on a phylogeny from a set of Procrustes shape variables

Usage

```
compare.multi.evol.rates(
  A,
  gp,
  phy,
  Subset = TRUE,
  iter = 999,
  seed = NULL,
  print.progress = TRUE
)
```

Arguments

A	A matrix (n x [p x k]) or 3D array (p x k x n) containing Procrustes shape variables for a set of specimens
gp	A factor array designating group membership for landmarks
phy	A phylogenetic tree of class phylo - see read.tree in library ape
Subset	A logical value indicating whether or not the traits are subsets from a single landmark configuration (default is TRUE)
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function compares net rates of morphological evolution for two or more multi-dimensional traits on a phylogeny, under a Brownian motion model of evolution following the procedure of Denton and Adams (2015). It is assumed that the landmarks for all traits have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. The approach calculates

multivariate net evolutionary rates found from the outer-product matrix of between species differences in morphospace after phylogenetic transformation (sensu Adams 2014). From the data the net rate of shape evolution for each multi-dimensional trait is calculated, and a ratio of rates is obtained. If three or more traits are used, the ratio of the maximum to minimum rate is used as a test statistic (see Denton and Adams 2015). Significance testing is accomplished by phylogenetic simulation in which tips data are obtained under Brownian motion using a an evolutionary rate matrix for all traits, which contains a common rate for all trait dimensions (Denton and Adams 2015). If three or more traits are used, pairwise p-values are also returned. In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019). Values from these distributions are log-transformed prior to effect size estimation, to assure normally distributed data.

The shape data may be input as either a 3D array (p x k x n) containing Procrustes shape variables for a set of species, or as a matrix (n x [p x k]) whose rows correspond to each species. In both cases, species names must be provided as rownames (for a matrix) or as the names of the third dimension of the array. Landmark groups for each trait are then specified by a factor array designating which landmark belongs to which trait. Additionally, if the method is to be used with other data (i.e., a set of length measurements), the input A should be a matrix of n rows of species and p columns of variables. In this case, the grouping factor should have each variable assigned to a trait group.

Comparisons of net evolutionary rates between traits may be accomplished in one of two ways. First, if the traits are are part of a single shape that was subjected to a single Procrustes superimposition (i.e., they are subsets of landmarks in the configuration), then the procedure is performed without alteration as described above. However, if the shapes are derived from different structures (shapes) that were superimposed separately, then the estimates of the rates must take the difference in the number of trait dimensions into account (see discussion in Denton and Adams 2015). This option is identified by selecting Subset = FALSE.

With `compare.multi.evol.rates`, the generic functions `print`, `summary`, and `plot` all work. The generic function, `plot`, produces a histogram of random rate-ratios associated with the resampling procedure.

Value

An object of class "evolrate" returns a list of the following:

`rates.all`	The phylogenetic evolutionary rates for each trait.
`rate.ratio`	The ratio of maximum to minimum evolutionary rates.
`pvalue`	The significance level of the observed rate ratio.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`pvalue.gps`	Matrix of pairwise significance levels comparing each pair of rates.
`call`	The matched call.

Author(s)

Dean Adams (used in some internal computations)

References

Adams, D.C. 2014. Quantifying and comparing phylogenetic evolutionary rates for shape and other high-dimensional phenotypic data. Syst. Biol. 63:166-177.

Denton, J.S.S., and D.C. Adams. 2015. A new phylogenetic test for comparing multiple high-dimensional evolutionary rates suggests interplay of evolutionary rates and modularity in lanternfishes (Myctophiformes; Myctophidae). Evolution. 69:2425-2440.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

Examples

```
data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment
land.gp<-c("A","A","A","A","A","B","B","B","B","B","B")
    #mandible and cranium subsets

EMR<-compare.multi.evol.rates(A=Y.gpa$coords,gp=land.gp,
    Subset=TRUE, phy= plethspecies$phy,iter=999)
summary(EMR)
plot(EMR)
```

---

compare.pls *Comparisons of Effect Sizes from Partial Least Squares*

Description

Function performs an analysis to compare the effect sizes of two or more PLS effects

Usage

```
compare.pls(..., two.tailed = TRUE)
```

Arguments

`...`	saved analyses of class pls
`two.tailed`	A logical value to indicate whether a two-tailed test (typical and default) should be performed.

Details

The function statistically compares the effect sizes of two or more PLS analyses. Typically, this function might be used to compare levels of integration between two or more samples, each measuring morphological integration between different modules. In such cases, the PLS correlation coefficient, r, is not a good measure of integration effect, as its expected value is dependent on both

the number of specimens and number of variables (Adams and Collyer 2016). This analysis calculates effect sizes as standard deviates, z, and performs two-sample z-tests, using the pooled standard error from the sampling distributions of the PLS analyses.

To use this function, perform `two.b.pls`, `integration.test`, or `phylo.integration` on as many samples as desired. Any number of objects of class pls can be input.

Similar versions of this function will be designed for alternative test statistics, in the future.

Notes for geomorph 3.0.4 and subsequent versions:

Compared to previous versions of geomorph, users might notice differences in effect sizes. Previous versions used z-scores calculated with expected values of statistics from null hypotheses (sensu Collyer et al. 2015); however Adams and Collyer (2016) showed that expected values for some statistics can vary with sample size and variable number, and recommended finding the expected value, empirically, as the mean from the set of random outcomes. Geomorph 3.0.4 and subsequent versions now center z-scores on their empirically estimated expected values and where appropriate, log-transform values to assure statistics are normally distributed. This can result in negative effect sizes, when statistics are smaller than expected compared to the average random outcome. For ANOVA-based functions, the option to choose among different statistics to measure effect size is now a function argument.

Value

An object of class compare.pls, returns a list of the following

`sample.z`	A vector of effect sizes for each sample.
`sample.r.sd`	A vector of standard deviations for each sampling distribution (following Box-Cox transformation).
`pairwise.z`	A matrix of pairwise, two-sample z scores between all pairs of effect sizes.
`pairwise.p`	A matrix of corresponding P-values.

Author(s)

Michael Collyer

References

Collyer, M.L., D.J. Sekora, and D.C. Adams. 2015. A method for analysis of phenotypic change for phenotypes described by high-dimensional data. Heredity. 115:357-365.

Adams, D.C. and M.L. Collyer. 2016. On the comparison of the strength of morphological integration across morphometric datasets. Evolution. 70:2623-2631.

Examples

```
# Example of comparative morphological integration between pupfish head
# and body shapes

 data(pupfish) # GPA previously performed

 group <- factor(paste(pupfish$Pop, pupfish$Sex, sep = "."))
```

```
levels(group)

tail.LM <- c(1:3, 5:9, 18:38)
head.LM <- (1:56)[-tail.LM]

tail.coords <- pupfish$coords[tail.LM,,]
head.coords <- pupfish$coords[head.LM,,]

# Subset 3D array by group, returning a list of 3D arrays
tail.coords.gp <- coords.subset(tail.coords, group)
head.coords.gp <- coords.subset(head.coords, group)

integ.tests <- Map(function(x,y) integration.test(x, y, iter=499,
print.progress = FALSE), head.coords.gp, tail.coords.gp)
# the map function performs the integration test on each 3D array in
# the lists provided

integ.tests$Marsh.F
integ.tests$Marsh.M
integ.tests$Sinkhole.F
integ.tests$Sinkhole.M

group.Z <- compare.pls(integ.tests)
summary(group.Z)

# Sexual dimorphism in morphological integration in one population
# but not the other

# can also list different PLS analyses, separately

compare.pls(MF = integ.tests$Marsh.F, MM = integ.tests$Marsh.M)
```

`coords.subset` *Subset landmark coordinates via a factor*

Description

Subset (split) landmark coordinates via a grouping factor

Usage

```
coords.subset(A, group)
```

Arguments

`A`	A 3D array (p x k x n) containing landmark coordinates for a set of specimens
`group`	A grouping factor of length n, for splitting the array into sub-arrays

Details

This function splits a set of landmark coordinates into subsets, as described by a factor. The result is a list of separate sets of landmarks.

Author(s)

Michael Collyer

Examples

```
data(pupfish)
group <- factor(paste(pupfish$Pop, pupfish$Sex))
levels(group)
new.coords <- coords.subset(A = pupfish$coords, group = group)
names(new.coords) # see the list levels
# group shape means
lapply(new.coords, mshape)
```

---

define.links *Define links between landmarks*

Description

An interactive function to define which landmarks should be linked to aid visualization.

Usage

```
define.links(spec, ptsize = 1, links = NULL)
```

Arguments

spec	Name of specimen, as an object matrix containing 2D or 3D landmark coordinates
ptsize	Numeric Size to plot the landmarks
links	Optional An existing links matrix to add on to

Details

Function takes a matrix of digitized landmark coordinates (e.g. from mshape) and allows the user to define pairs of landmarks to be linked, for visualization purposes. The output is a matrix to be used by plotAllSpecimens & plotRefToTarget option 'links='.

Selection:

In the plot window select two landmarks that will be linked. In the console, the user will be prompted to continue linking landmarks to build the wireframe or end the session (typing y or n respectively). For 2D data in plot window, use LEFT mouse button to select landmarks. For 3D data in rgl window, use RIGHT mouse button (or command+LEFT for mac) to select landmarks.

Value

Function returns a matrix of which landmarks will be links to be used by plotAllSpecimens & plotRefToTarget option 'links='.

Author(s)

Emma Sherratt

See Also

plotAllSpecimens

plotRefToTarget

rgl-package (used in 3D plotting)

---

define.modules *Define modules (landmark partitions)*

Description

An interactive function to define which landmarks should be assigned to each module (landmark partition).

Usage

```
define.modules(spec, nmodules)
```

Arguments

spec	A p x k matrix containing landmark coordinates of a single specimen (2D or 3D)
nmodules	Number of modules to be defined

Details

Function takes a matrix of digitized landmark coordinates (e.g. from mshape) and allows the user to assign landmarks to each module. The output is a list of which landmarks belong in which partition, to be used by modularity.test or integration.test.

Selection in 2D:

Choosing which landmarks will be included in each module involves landmark selection using a mouse in the plot window. The user is prompted to select each landmark in turn to be assigned to module 1: using the LEFT mouse button (or regular button for Mac users), click on the hollow circle to choose the landmark. Selected landmarks will be filled in. When all landmarks for module 1 are chosen, press 'esc', and then start selecting landmarks for module 2. Repeat until all modules are defined.

Selection in 3D:

Choosing which landmarks will be included in each module involves landmark selection using a mouse in the rgl plot window. The user is prompted to select one or more landmarks. To do so, use the RIGHT mouse button (or command + LEFT button for Mac users), draw a rectangle around landmarks to select. Selected landmarks will be colored yellow. Then type into the console a letter (e.g. 1, 2, 3...) to assign selected landmark(s) to this module. Repeat until all landmarks are assigned to modules.

Value

Function returns a vector of which landmarks belong in which module (e.g. 1, 1, 1, 2, 2, 3, 3, 3, 2) to be used with `modularity.test` or `integration.test`.

Author(s)

Emma Sherratt

See Also

`modularity.test` and `integration.test`

`rgl-package` (used in 3D plotting)

`define.sliders` *Select points to "slide" along curves*

Description

An interactive function to define which landmarks will "slide" along two-dimensional (2D) or three-dimensional (3D) curves.

Usage

```
define.sliders(landmarks, nsliders, surfsliders = NULL, write.file = TRUE)
```

Arguments

`landmarks`	A matrix containing 2D or 3D landmark coordinates of landmarks and semi-landmarks, OR A vector containing a sequence of numbers corresponding to the landmarks in the order they appear along the curve (for AUTO mode)
`nsliders`	Number of landmarks to be semilandmarks that slide along curves
`surfsliders`	(3D only) If 'landmarks' contains "surface sliders", these should be given as a vector or use surfsliders = T, and function looks for "surfslide.csv" in working directory.
`write.file`	A logical value indicating whether the matrix is written to file as .csv.

Details

Function takes a matrix of digitized landmark coordinates, such as made by digitize2d or digit.fixed, and helps user choose which landmarks will be treated as "sliders" in Generalized Procrustes analysis gpagen. This type of semilandmark "slides" along curves lacking known landmarks (see Bookstein 1997 for algorithm details). Each sliding semilandmark ("sliders") will slide between two designated points, along a line tangent to the specified curvature.

Defining landmarks is an interactive procedure (see below for 2D and 3D routines). The procedure is overlapping. For example: there are 5 landmarks (1:5), 1 and 5 are landmarks and 2,3,4 are sliders, the user must select '1' '2' '3', and then '2' '3' '4', and then '3' '4' '5'.

Selection in 2D:

Choosing which landmarks will be sliders involves landmark selection using a mouse in the plot window. To define the sliders, for each sliding landmark along the curve in the format 'before-slider-after', using the LEFT mouse button (or regular button for Mac users), click on the hollow circle to choose the landmark in the following order:

1. Click to choose the first landmark between which semi-landmark will "slide",
2. Click to choose sliding landmark,
3. Click to choose the last landmark between which semi-landmark will "slide", Selected landmarks will be filled in and lines are drawn connecting the three landmarks, and will highlight the sliding semilandmark in red and the flanking landmarks in blue.

Selection in 3D:

Choosing which landmarks will be sliders involves landmark selection using a mouse in the rgl plot window. With a standard 3-button (PC) buildtemplate uses:

1. the RIGHT mouse button (primary) to choose points to be defined as sliders,
2. the LEFT mouse button (secondary) is used to rotate mesh,
3. the mouse SCROLLER (third/middle) is used to zoom in and out.

NOTE: Digitizing functions on MACINTOSH computers using a standard 3-button mice works as specified. Macs using platform specific single button mice, XQuartz must be configured: go to Preferences > Input > tick "Emulate three button mouse":

1. press button to rotate 3D mesh,
2. press button while pressing COMMAND key to select points to be defined as sliders,
3. press button while pressing OPTION key to adjust mesh perspective.
4. the mouse SCROLLER or trackpad two finger scroll is used to zoom in an out.

To define the sliders, for each sliding landmark along the curve in the format 'before-slider-after':

1. Click to choose the first landmark between which semi-landmark will "slide",
2. Click to choose sliding landmark,
3. Click to choose the last landmark between which semi-landmark will "slide", Screen will show lines connecting the three landmarks, and will highlight the sliding semilandmark in red.

AUTO mode:

The input 'landmarks' can be simply a vector of numbers corresponding to the "sliders" (semi-landmarks) in the order they appear along a curve on the specimen. This can be made by c() or seq() or any other reasonable method.

If the sliders form a closed curve, then the function assumes that the first and last landmarks in the 'landmarks' vector are THE SAME are fixed (not sliders). e.g. if landmark 1 is a fixed landmark, and 2, 3 and 4 are semilandmarks, then sliders = c(1,2,3,4,1).

Value

Function returns a 'nsliders-x-3' matrix containing the landmark address of the curve sliders, indicating the landmarks between which the slider landmarks will "slide". If write.file = T the matrix is also written to working directory as "curveslide.csv". Matrix (or "curveslide.csv") is designed for use by gpagen during GPA.

Author(s)

Emma Sherratt & Dean Adams

References

Bookstein, F. J. 1997 Landmark Methods for Forms without Landmarks: Morphometrics of Group Differences in Outline Shape. Medical Image Analysis 1(3):225-243.

See Also

digitize2d, digit.fixed, gpagen, digit.curves

rgl-package (used in 3D plotting)

Examples

```
## (not run) Use interactive function in rgl window
 # data(scallops)
 # define.sliders(scallops$coorddata[,,1], nsliders=11,
 #   surfsliders = scallops$surfslide)
 # here the first specimen is used for plotting purposes only

## Examples of AUTO mode
 ## 1 curve of sliding semilandmark
 # Define sliders for scallopdata
 #sliders = define.sliders(c(5:16,1))

 ## 2 curves of sliding semilandmarks
 # Define sliders for 10 landmarks, where LMs 1, 5, and 10 fixed
 # 2, 3, and 4 are along a curve between 1 and 5
 # and 6, 7, 8, and 9 are along a curve between 5 and 10.
 #sliders = rbind(define.sliders(1:5), define.sliders(5:10))
```

digit.curves *Calculate semilandmarks along a curve*

Description

A function that "digitizes curves" by calculating equidistant two-dimensional or three-dimensional semilandmarks along a curve. These landmarks will be treated as "sliders" in Generalized Procrustes analysis gpagen. This type of semilandmark "slides" along curves lacking known landmarks (see Bookstein 1997 for algorithm details). Each sliding semilandmark ("sliders") will slide between two designated points, along a line tangent to the specified curvature, as specified by define.sliders.

Usage

```
digit.curves(start, curve, nPoints, closed = TRUE)
```

Arguments

start	A numeric vector of x,y,(z) coordinates for the landmark defining the start of the curve (can be simply first point on open outline: curve[1,])
curve	A matrix (p x k) of 2D or 3D coordinates for a set of ordered points defining a curve
nPoints	Numeric how many semilandmarks to place equidistantly along the curve (not counting beginning and end points)
closed	Logical Whether the curve is closed (TRUE) or open (FALSE)

Details

The function is based upon tpsDig2 'resample curve by length' for 2D data by James Rohlf (Rohlf 2015). The start of the curve is a fixed landmark on the curve that is equivalent (homologous) in each specimen in the sample (and will be treated as a fixed point during Procrustes Superimposition using gpagen). Then nPoints are calculated along the curve at equidistant points from the start to the end.

'curve' is a p-x-k matrix of 2D or 3D coordinates for a set of ordered points defining a curve. This can be the pixels of an outline calculated in ImageJ (save xy coordinates) or any other reasonable way of obtaining ordered coordinates along a curve (including sampling by hand using digit.fixed or digitize2d - but note that there should be more points defining the curve than nPoints in order to accurately calculate the semilandmarks).

If 'closed = T', the function returns the coordinates of the 'start' landmark plus nPoints. If 'closed = F', the function returns the coordinates of the 'start' landmark, plus nPoints and the end of the curve.

If unsure if the points defining the curve are ordered, then plot and color them using the rainbow function, e.g. plot(curve, pch=19, cex=0.1, col=rainbow(nrow(outline))), and it should be easy to visualize.

Value

Function returns a matrix of coordinates for nPoints equally spaced semilandmarks sampled along the curve (plus start and end if 'closed = F', or only including start if 'closed = T')

Author(s)

Emma Sherratt and Michael Collyer

References

Bookstein, F. J. 1997 Landmark Methods for Forms without Landmarks: Morphometrics of Group Differences in Outline Shape. Medical Image Analysis 1(3):225-243.

Rohlf, F.J., 2015. The tps series of software. Hystrix 26(1):9-12.

See Also

digit.fixed digitize2d

---

digit.fixed *Digitize 3D landmarks on mesh3d object*

Description

An interactive function to digitize three-dimensional (3D) landmarks. Input for the function is either a matrix of vertex coordinates defining a 3D surface object or a mesh3d object as obtained from read.ply.

Usage

```
digit.fixed(spec, fixed, index = FALSE, ptsize = 1, center = TRUE)
```

Arguments

spec	An object of class shape3d/mesh3d, or matrix of 3D vertex coordinates
fixed	The number of landmarks to be digitized (fixed, and curve sliders if desired)
index	Whether selected landmark addresses should be returned (internal use only)
ptsize	Size of mesh points (vertices), e.g. 0.1 for dense meshes, 3 for sparse meshes
center	Should the object 'spec' be centered prior to digitizing?

Details

Function for digitizing fixed three-dimensional landmarks. The user can later designate some as curve sliding semilandmarks, using the function `define.sliders` or through a semilandmark definition matrix.

To digitize 3D surface sliding semilandmarks the function `digitsurface` should be used instead.

For details on the full procedure for digitizing fixed 3D landmarks and surface sliding semilandmarks, see the relevant vignette by running `vignette("geomorph.digitize3D")`.

NOTE: The function centers the mesh before digitizing by default (center=TRUE). If one chooses not to center, specimen may be difficult to manipulate in rgl window.

Digitizing:

Digitizing is interactive. Once a point is selected, the user is asked if the system should keep or discard the selection (y/n). If "y", the user is asked to continue to select the next landmark. If "n" the removes the last chosen landmark, and the user is asked to select it again. This can be repeated until the user is comfortable with the landmark chosen.

To digitize with a standard 3-button mouse (PC):

1. the RIGHT mouse button (primary) to select points to be digitized,
2. the LEFT mouse button (secondary) is used to rotate mesh,
3. the mouse SCROLLER (third/middle) is used to zoom in and out.

NOTE: Digitizing functions on MACINTOSH computers using a standard 3-button mice works as specified. Macs using platform specific single button mice, XQuartz must be configured: go to Preferences > Input > tick "Emulate three button mouse":

1. press button to rotate 3D mesh,
2. press button while pressing COMMAND key to select vertex to be used as a landmark,
3. press button while pressing OPTION key to adjust mesh perspective.
4. the mouse SCROLLER or trackpad two finger scroll is used to zoom in an out.

NOTE: there is no pan (translate) functionality in rgl library for all platforms at this time.

Value

Function returns (if assigned to an object) and writes to the working directory an NTS file, containing the landmark coordinates. The file name corresponds to the name of the specimen. If index=FALSE function returns to the console an n x 3 matrix containing the x,y,z coordinates of the digitized landmarks. If index=TRUE, function returns a list:

`selected`	a matrix containing the x,y,z coordinates of the digitized landmarks
`fix`	a matrix of addresses for landmarks that are "fixed" (for internal use)

Author(s)

Erik Otarola-Castillo & Emma Sherratt

See Also

`read.ply`

`rgl-package` (used in 3D plotting)

digitize2d *Digitize 2D landmarks on .jpg files*

Description

An interactive function to digitize two-dimensional(2D) landmarks from .jpg files.

Usage

```
digitize2d(
  filelist,
  nlandmarks,
  scale = NULL,
  tpsfile,
  MultScale = FALSE,
  verbose = TRUE
)
```

Arguments

filelist	A list of names of jpeg images to be digitized.
nlandmarks	Number of landmarks to be digitized.
scale	A vector containing the length of the scale to be placed on each image.
tpsfile	The name of a TPS file to be created or read
MultScale	A logical option indicating if the coordinates should be pre-multiplied by scale
verbose	logical. User decides whether to digitize in verbose or silent format (see details), default is verbose

Details

This function may be used for digitizing 2D landmarks from jpeg images (.jpg). The user provides a list of image names, the number of landmarks to be digitized, and the name of an output TPS file. An option is included to allow the user to digitize a scale on each image to convert the landmark coordinates from pixels into meaningful units. Landmarks to be digitized can include both fixed landmarks and semi-landmarks, the latter of which are to be designated as "sliders" for subsequent analysis (see the function define.sliders).

The Digitizing Session: Digitizing landmarks from 2D photos requires that a scale bar is placed in the image in order to scale the coordinate data. The 'scale' option requires: a single number (e.g. 10) which means that the scale to be measured in all images is a 10mm scale bar; OR a vector the same length as the filelist containing a number for the scale of each image. If scale=NULL, then the digitized coordinates will not be scaled. This option is NOT recommended.

Users may digitize all specimens in one session, or may return at a later time to complete digitizing. In the latter case, the user provides the same filelist and TPS file and the function will determine where the user left off.

If specimens have missing landmarks, these can be incorporated during the digitizing process using the 'a' option as described below (a=absent).

Specimen Digitizing:

Digitizing landmarks involves landmark selection using a mouse in the plot window, using the LEFT mouse button (or regular button for Mac users):

1. Digitize the scale bar (if requested) by selecting the two end points. Use a single click for start and end points. The user is asked whether the system should keep or discard the digitized scale bar.
2. Digitize each landmark with single click and the landmark is shown in red.

If verbose = TRUE, digitizing is interactive between landmark selection using a mouse and the R console. Once a landmark is selected, the user is asked if the system should keep or discard the selection (y/n/a). If "y", the user is asked to continue to select the next landmark. If "n", the user is asked to select it again.

To digitize a missing landmark, simply click on any location in the image. Then, when prompted to keep selection, choose 'a' (for absent). Missing landmarks can only be included during the digitizing process when verbose=TRUE.

If verbose = FALSE the digitizing of landmarks is continuous and uninterrupted. Here the user will not be prompted to approve each landmark selection.

At the end of digitizing, the landmark coordinates are written to a TPS file. By default, the x,y values are unscaled if a vector of scales is included, and the scale is returned on line SCALE= after each specimen x,y data. Optionally, one may have the coordinates pre-multiplied by scale by using the option MultScale=TRUE.

Value

Function returns a tps file containing the digitized landmark coordinates.

Author(s)

Dean Adams, Erik Otarola-Castillo and Emma Sherratt

See Also

`readJPEG` (for JPEG input)

`digitsurface` *Digitize 3D fixed landmarks and surface semilandmarks*

Description

An interactive function to digitize three-dimensional (3D) landmarks on a surface lacking known landmarks. Input for the function is either a matrix of vertex coordinates defining a 3D surface object or a mesh3d object as obtained from `read.ply`.

Usage

```
digitsurface(spec, fixed, ptsize = 1, center = TRUE)
```

Arguments

spec	An object of class shape3d/mesh3d, or matrix of 3D vertex coordinates
fixed	Either a numeric value designating the number of fixed landmarks to be selected by digit.fixed, or a matrix of 3D coordinates collected previously
ptsize	Size of mesh points (vertices), e.g. 0.1 for dense meshes, 3 for sparse meshes
center	Should the object 'spec' be centered prior to digitizing?

Details

Function for digitizing fixed 3D landmarks and placing surface sliding semilandmarks using a previously created template. Following the selection of fixed points (see digitizing below), the function finds surface semilandmarks following the algorithm outlined in Gunz et al. (2005) and Mitteroecker and Gunz (2009). digitsurface finds the same number of surface semilandmarks as the template (created by buildtemplate) by downsampling the scanned mesh, after registering the template with the current specimen via GPA. A nearest neighbor algorithm is used to match template surface semilandmarks to mesh points of the current specimen. To use function digitsurface, the template must be constructed first, and 'template.txt' be in the working directory. Because template matching is based on the correspondence of fixed landmark points in the template and the specimen, a minimum of four fixed landmarks must be used.

For details on the full procedure for digitizing fixed 3D landmarks and surface sliding semilandmarks, see the relevant vignette by running `vignette("geomorph.digitize3D")`.

NOTE: Function centers the mesh before digitizing by default (center=TRUE). If one chooses not to center, specimen may be difficult to manipulate in rgl window.

Digitizing: Digitizing of fixed landmarks is interactive. Once a point is selected, the user is asked if the system should keep or discard the selection (y/n). If "y", the user is asked to continue to select the next landmark. If "n" the removes the last chosen landmark, and the user is asked to select it again. This can be repeated until the user is comfortable with the landmark chosen.

To digitize with a standard 3-button (PC):

1. the RIGHT mouse button (primary) to select points to be digitized,
2. the LEFT mouse button (secondary) is used to rotate mesh,
3. the mouse SCROLLER (third/middle) is used to zoom in and out.

NOTE: Digitizing functions on MACINTOSH computers using a standard 3-button mice works as specified. Macs using platform specific single button mice, XQuartz must be configured: go to Preferences > Input > tick "Emulate three button mouse":

1. press button to rotate 3D mesh,
2. press button while pressing COMMAND key to select vertex to be used as a landmark,
3. press button while pressing OPTION key to adjust mesh perspective.
4. the mouse SCROLLER or trackpad two finger scroll is used to zoom in an out.

NOTE: there is no pan (translate) functionality in rgl library for all platforms at this time.

AUTO mode:

The function as described above (for interactive mode) calls digit.fixed, prompting the user to select fixed landmarks in the rgl window. However if the user has digitized these fixed landmark elsewhere (e.g., in other software), then the input for parameter 'fixed' can be a p-x-k matrix of 3D coordinates. In this case, the function the function will automatically use these landmarks and fit the template of sliding semilandmarks.

Value

Function returns (if assigned to an object) and writes to the working directory an NTS file, containing the landmark coordinates. The file name corresponds to the name of the specimen.

Author(s)

Erik Otarola-Castillo & Emma Sherratt

References

Gunz P, Mitteroecker P, & Bookstein FJ (2005) Semilandmarks in Three Dimensions. Modern Morphometrics in Physical Anthropology, ed Slice DE (Springer-Verlag, New York), pp 73-98.

Mitteroecker P & Gunz P (2009) Advances in Geometric Morphometrics. Evolutionary Biology 36(2):235-247.

See Also

buildtemplate

read.ply

digit.fixed

rgl-package (used in 3D plotting)

---

editTemplate *Edit 3D template*

Description

An interactive function to remove landmarks from a 3D template file.

Usage

```
editTemplate(template, fixed, n)
```

Arguments

template	Matrix of template 3D coordinates
fixed	Number of "fixed" landmark points (non surface sliding points)
n	Number of points to be removed

Details

The function edits a 'template.txt' file made by buildtemplate, which must be in the current working directory, and which is overwritten. Use read.table("template.txt", header = T) to read in the template first.

Selection:

Choosing which landmarks will be deleted involves landmark selection using a mouse in the rgl plot window. With a standard 3-button (PC) buildtemplate uses:

1. the RIGHT mouse button (primary) to choose points to be deleted,
2. the LEFT mouse button (secondary) is used to rotate mesh,
3. the mouse SCROLLER (third/middle) is used to zoom in and out.

NOTE: Digitizing functions on MACINTOSH computers using a standard 3-button mice works as specified. Macs using platform specific single button mice, XQuartz must be configured: go to Preferences > Input > tick "Emulate three button mouse":

1. press button to rotate 3D mesh,
2. press button while pressing COMMAND key to select points to be deleted,
3. press button while pressing OPTION key to adjust mesh perspective.
4. the mouse SCROLLER or trackpad two finger scroll is used to zoom in an out.

Value

Function returns a matrix containing the x,y,z coordinates of the new template landmarks. Function also writes to the working directory 'template.txt' containing the x,y,z coordinates of the updated template

Author(s)

Erik Otarola-Castillo & Emma Sherratt

See Also

rgl-package (used in 3D plotting)

`estimate.missing` *Estimate locations of missing landmarks*

Description

A function for estimating the locations of missing landmarks

Usage

```
estimate.missing(A, method = c("TPS", "Reg"))
```

Arguments

A	An array (p x k x n) containing landmark coordinates for a set of specimens or a geomorphShapes object
method	Method for estimating missing landmark locations

Details

The function estimates the locations of missing landmarks for incomplete specimens in a set of landmark configurations, where missing landmarks in the incomplete specimens are designated by NA in place of the x,y,z coordinates. Two distinct approaches are implemented.

The first approach (method="TPS") uses the thin-plate spline to interpolate landmarks on a reference specimen to estimate the locations of missing landmarks on a target specimen. Here, a reference specimen is obtained from the set of specimens for which all landmarks are present, Next, each incomplete specimen is aligned to the reference using the set of landmarks common to both. Finally, the thin-plate spline is used to estimate the locations of the missing landmarks in the target specimen (Gunz et al. 2009).

The second approach (method="Reg") is multivariate regression. Here each landmark with missing values is regressed on all other landmarks for the set of complete specimens, and the missing landmark values are then predicted by this linear regression model. Because the number of variables can exceed the number of specimens, the regression is implemented on scores along the first set of PLS axes for the complete and incomplete blocks of landmarks (see Gunz et al. 2009). Note, however, that a minimum of k*m+k specimens are required to estimate m missing landmarks (of k-dimension) in any one specimen using the regression method. More generally, if the number of missing landmarks approaches the number of reference specimens used to estimate them, estimation will become increasingly imprecise with the regression method. Additionally, the location of missing landmarks (contiguous versus disparate in location) can also influence the precision of estimation. The user should be aware that the function will produce results but the results from the regression method might be influenced by the number of specimens, the number of total landmarks, and the number and location of missing landmarks in any one specimen. It might be wise to compare multiple methods for specific cases, if uncertain about the precision of estimation.

One can also exploit bilateral symmetry to estimate the locations of missing landmarks. Several possibilities exist for implementing this approach (see Gunz et al. 2009). Example R code for one implementation is found in Claude (2008).

NOTE: Because all geometric morphometric analyses and plotting functions implemented in geomorph require a full complement of landmark coordinates, the alternative to estimating the missing landmark coordinates is to proceed with subsequent analyses EXCLUDING specimens with missing values.

Value

Function returns an array (p x k x n) of the same dimensions as input A, including coordinates for the target specimens (the original landmarks plus the estimated coordinates for the missing landmarks). If the input is a geomorphShapes object, this is returned with the original and estimated coordinates in $landmarks In both cases, these data need to be Procrustes Superimposed prior to analysis, including sliding of semilandmarks (see gpagen).

Author(s)

Dean Adams

References

Claude, J. 2008. Morphometrics with R. Springer, New York.

Bookstein, F. L., K. Schafer, H. Prossinger, H. Seidler, M. Fieder, G. Stringer, G. W. Weber, J.-L. Arsuaga, D. E. Slice, F. J. Rohlf, W. Recheis, A. J. Mariam, and L. F. Marcus. 1999. Comparing frontal cranial profiles in archaic and modern Homo by morphometric analysis. Anat. Rec. (New Anat.) 257:217-224.

Gunz, P., P. Mitteroecker, S. Neubauer, G. W. Weber, and F. L. Bookstein. 2009. Principles for the virtual reconstruction of hominin crania. J. Hum. Evol. 57:48-62.

Examples

```
data(plethodon)
plethland<-plethodon$land
  plethland[3,,2]<-plethland[8,,2]<-NA  #create missing landmarks
  plethland[3,,5]<-plethland[8,,5]<-plethland[9,,5]<-NA
  plethland[3,,10]<-NA

estimate.missing(plethland,method="TPS")
estimate.missing(plethland,method="Reg")
```

`findMeanSpec` *Identify specimen closest to the mean of a set of Procrustes shape variables*

Description

A function to identify which specimen lies closest to the estimated mean shape for a set of Procrustes shape variables.

Usage

```
findMeanSpec(A)
```

Arguments

`A`	A 3D array (p x k x n) containing landmark coordinates for a set of Procrustes shape variables

Details

Function takes a 3D array of Procrustes shape variables (such as made by `gpagen`, calculates the distance of each to the estimated mean shape, and returns the name and address of the closest specimen. This function can be used to identify the specimen to be used by `warpRefMesh`.

Value

Function returns the name and address of the specimen closest to the mean of the set of Procrustes shape variables.

Author(s)

Emma Sherratt

See Also

warpRefMesh

`fixed.angle` *Rotate a subset of 2D landmarks to common articulation angle*

Description

A function for rotating a subset of landmarks so that the articulation angle between subsets is constant

Usage

```
fixed.angle(
  A,
  art.pt = NULL,
  angle.pts.1,
  angle.pts.2,
  rot.pts = NULL,
  angle = 0,
  degrees = FALSE
)
```

Arguments

`A`	A 3D array (p x k x n) containing landmark coordinates for a set of specimens
`art.pt`	A number specifying which landmark is the articulation point between the two landmark subsets
`angle.pts.1`	A vector or single value specifying the angle point of one subset. If more that one value is provided, the centroid of the landmarks described by the vector will be used; a single value identifies a specific landmark to use.
`angle.pts.2`	A vector or single value specifying the angle point of the second subset. This could be the entire set of points of an articulated structure to be rotated.
`rot.pts`	A vector containing numbers specifying which landmarks are in the subset to be rotated. If NULL, it is assumed that the points to be rotated are the same as those in angle.pts.2.

angle	An optional value specifying the additional amount by which the rotation should be augmented (in radians). It might be essential to use a negative angle if centroids from multiple points are used for angle points. It should be clear if this is the case, upon plotting results.
degrees	A logical value specifying whether the additional rotation angle is expressed in degrees or radians (radians is default)

Details

This function standardizes the angle between two subsets of landmarks for a set of specimens. The approach assumes a simple hinge-point articulation between the two subsets, and rotates all specimens such that the angle between landmark subsets is equal across specimens (see Adams 1999). As a default, the mean angle is used, though the user may specify an additional amount by which this may be augmented. To quantify the angle, users may specify a single landmark in each subset as angle endpoints, or may specify a set of landmarks. If the latter, the centroid of those points is used.

Presently, the function is only implemented for two-dimensional landmark data.

Value

Function returns a (p x k x n) array of landmark coordinates.

Author(s)

Dean Adams and Michael Collyer

References

Adams, D. C. 1999. Methods for shape analysis of landmark data from articulated structures. Evolutionary Ecology Research. 1:959-970.

Examples

```
#Example using Plethodon
#Articulation point is landmark 1, rotate mandibular landmarks (2-5)
# relative to cranium

data(plethspecies)
# Using specific points:
newLM1 <- fixed.angle(plethspecies$land,
art.pt=1, angle.pts.1 = 5,
angle.pts.2 = 6, rot.pts = c(2,3,4,5))
Y.gpa1 <- gpagen(newLM1)
plot(Y.gpa1, mean = FALSE)

# Using centroids from subsets
newLM2 <- fixed.angle(plethspecies$land,art.pt=1,
angle.pts.1 = c(1, 6:11),
angle.pts.2 = 2:5,
rot.pts = NULL, angle = 20,
```

```
degrees = TRUE) # rotated points same as second partition
Y.gpa2 <- gpagen(newLM2)
plot(Y.gpa2, mean = FALSE)
```

`geomorph.data.frame` *Create a data frame with shape data*

Description

A list similar to a data frame to facilitate analysis of shape data.

Usage

```
geomorph.data.frame(...)
```

Arguments

...	a list of objects to include in the data frame.

Details

This function produces a list that can be used like a data frame in other analytical functions. The purpose is similar to the function, `data.frame`, but without the constraint that data must conform to an n (observations) x p (variables) matrix. Rather, the list produced is constrained only by n. List objects can be Procrustes shape variables, matrices, variables, distance matrices, and phylogenetic trees. Results from `gpagen` can be directly imported into a geomorph.data.frame to utilize the coordinates and centroid size as variables. (See Examples)

Author(s)

Michael Collyer

Examples

```
data(plethodon)
Y.gpa <- gpagen(plethodon$land,PrinAxes=FALSE)
gdf <- geomorph.data.frame(Y.gpa)
attributes(gdf)

gdf <- geomorph.data.frame(Y.gpa, species = plethodon$species,
site = plethodon$site)
attributes(gdf)

# Using geomorph.data.frame to facilitate analysis
anova(procD.lm(coords ~ Csize + species * site, data = gdf))
```

`globalIntegration` *Quantify global integration relative to self-similarity*

Description

Function quantifies the overall level of morphological integration for a set of Procrustes shape variables

Usage

```
globalIntegration(A, ShowPlot = TRUE)
```

Arguments

`A`	3D array (p1 x k x n) containing Procrustes shape variables
`ShowPlot`	A logical value indicating whether or not the plot should be returned

Details

The function quantifies the overall level of morphological integration for a set of Procrustes shape coordinates. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. Based on the set of aligned specimens, the function estimates the set of bending energies at various spatial scales, and plots the log of the variance of the partial warps versus the log of their corresponding bending energies (Bookstein 2015). The slope of a regression of these data provides information regarding the degree of overall morphological integration (or lack thereof).

A slope of negative one corresponds to self-similarity, implying that patterns of shape variation are similar across spatial scales. Steeper slopes (i.e., those more extreme than -1.0) correspond to data that are globally integrated, while shallower slopes (between -1 and 0) correspond to data that are 'disintegrated (see Bookstein 2015). Isotropic data will have an expected slope of zero.

Author(s)

Dean Adams

References

Bookstein, F. L. 2015. Integration, disintegration, and self-similarity: Characterizing the scales of shape variation in landmark data. Evol. Biol.42(4): 395-426.

Examples

```
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment

globalIntegration(Y.gpa$coords)
```

`gm.prcomp` *Principal and phylogenetically-aligned components analysis of shape data*

Description

Function performs principal components analysis (PCA) or phylogenetically-aligned components (PaCA) on Procrustes shape coordinates.

Usage

```
gm.prcomp(
  A,
  phy = NULL,
  align.to.phy = FALSE,
  GLS = FALSE,
  transform = FALSE,
  ...
)
```

Arguments

`A`	A 3D array (p x k x n) containing Procrustes shape variables for a set of aligned specimens. Alternatively, this can be an n x p matrix of any data, but output will not contain information about shapes.
`phy`	An optional phylogenetic tree of class phylo
`align.to.phy`	An optional argument for whether **PaCA** (if TRUE) should be performed
`GLS`	Whether GLS-centering and covariance estimation should be used (rather than OLS).
`transform`	A logical value to indicate if transformed residuals should be projected. This is only applicable if GLS = TRUE. If TRUE, an orthogonal projection of transformed data is made; if FALSE an oblique projection of untransformed data is made.
`...`	Other arguments passed to `ordinate` and `scale`. The most common arguments are scale., tol, and rank.

Details

The function performs a series of ordinations, taking into account, phylogeny, if desired. There are two main types of ordinations: principal components analysis (PCA) and phylogenetically-aligned components analysis (PaCA). Both of these have two variants: centering and projection via ordinary least-squares (OLS) or via generalized least-squares (GLS). The name, "gm.prcomp", references that this function performs much like `prcomp`, in terms of arguments and output, but this function is quite a bit more diverse. This function has the capability of performing analyses generally referred to as:

- **PCA** Standard PCA based on OLS-centering and projection of data.
- **Phylomorphospace** Standard PCA with estimated ancestral states and phylogenetic branches projected into ordination plots.
- **phyloPCA** PCA based on GLS-centering and projection of data. Also possible to project ancestral states into plots. Note that if transformed GLS-residuals are used for projection, the ancestral states might not appear logical, as the projection is independent of phylogeny. With OLS-centering, a phyloPCA as described by Revell (2009) is produced.
- **PaCA** Phylogenetically-aligned component analysis. Data are aligned to an axis of greatest phylogenetic signal rather than axis of greatest dispersion. This analysis can use either OLS- or GLS-centering and projection. Phylogenetic signal is strongest in the first few components of the OLS approach. This analysis will make little sense with GLS-centering and projection of transformed residuals, since phylogenetic signal is removed the transformed data. See Collyer and Adams (2021) for more details. For greater flexibility for type of residuals and projection of trees, use `ordinate`. See Collyer and Adams (2021) for details.

- **phy** Whether a phylogeny and estimated ancestral states are considered in plots.
- **align.to.phy** Whether components are aligned to phylogenetic signal (rather than principal axes).
- **GLS** Whether to use GLS-centering and estimation of covariance matrix.
- **transform** Whether to transform GLS-residuals (making them independent of phylogeny and an orthogonal projection from the transformed data space, as opposed to an oblique projection from the untransformed data space).

PLOTTING: Contrary to previous geomorph implementations, gm.prcomp does not produce plots. For plotting gm.prcomp class objects combine `plot.gm.prcomp` and `picknplot.shape` following the examples below.

SUMMARY STATISTICS: For principal component plots, the traditional statistics to summarize the analysis include eigenvalues (variance by component), proportion of variance by component, and cumulative proportion of variance. When data are aligned to a phylogenetic covariance matrix, the statistics are less straightforward. A summary of of such an analysis (performed with `summary.gm.prcomp`) will produce these additional statistics:

- **Singular Value** Rather than eigenvalues, the singular values from singular value decomposition of the cross-product of the scaled phylogenetic covariance matrix and the data.
- **Proportion of Covariance** Each component's singular value divided by the sum of singular values. The cumulative proportion is also returned. Note that these values do not explain the amount of covariance between phylogeny and data, but explain the distribution of the covariance. Large proportions can be misleading.
- **RV by Component** The partial RV statistic by component. Cumulative values are also returned. The sum of partial RVs is Escoffier's RV statistic, which measures the amount of covariation between phylogeny and data. Caution should be used in interpreting these values, which can vary with the number of observations and number of variables. However, the RV is more reliable than proportion of singular value for interpretation of the strength of linear association for phylogenetically-aligned components.
- **Tips or Ancestors Dispersion** The variances of points by component for tip data and estimated ancestral character states, after projection. These values will differ from variances

from PCA with GLS estimation, as the "Importance of Components" weights variances by phylogenetic covariances. Dispersion statistics correspond to the amount of scatter in plots of component scores.

NOTE: The plot.gm.prcomp function performs the same plotting that was previously possible with plotTangentSpace and plotGMPhyloMorphoSpace, which have now been deprecated.

Value

An object of class "gm.prcomp" contains a list of results for each of the PCA approaches implemented. Each of these lists includes the following components:

x	Component scores for all specimens.
anc.x	Component scores for the ancestors on the phylogeny.
d	The singular values of the decomposed VCV matrix.
rotation	The matrix of variable loadings, i.e. the eigenvectors of the decomposed matrix.
shapes	A list with the shape coordinates of the extreme ends of all PC axes.
ancestors	The matrix of estimated ancestral shapes, if a phylogeny is used.
anc.var	The variances among ancestor scores, by component, if a phylogeny is used.

Author(s)

Antigoni Kaliontzopoulou, Michael Collyer, & Dean Adams

References

Collyer, M.L and D.C. Adams, 2021. Phylogenetically Aligned component analysis. Methods in Ecology and Evolution, 12: 369-372.

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

See Also

plot.gm.prcomp

picknplot.shape

ordinate A more bare-bones ordination function on which gm.prcomp depends.

Examples

```
data(plethspecies)
Y.gpa <- gpagen(plethspecies$land)    #GPA-alignment

###  Traditional PCA
PCA <- gm.prcomp(Y.gpa$coords)
summary(PCA)
plot(PCA, main = "PCA")
plot(PCA, main = "PCA", flip = 1) # flip the first axis
plot(PCA, main = "PCA", axis1 = 3, axis2 = 4) # change PCs viewed
```

```
### Phylomorphospace - PCA with phylogeny (result is same as above,
### but with estimated ancestral states projected into plot)
PCA.w.phylo <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy)
summary(PCA.w.phylo)
plot(PCA.w.phylo, phylo = TRUE, main = "PCA.w.phylo")

### Phylogenetic PCA - PCA based on GLS-centering and projection
# This is the same as the method described by Revell (2009)
phylo.PCA <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy, GLS = TRUE)
summary(phylo.PCA)
plot(phylo.PCA, phylo = TRUE, main = "phylo PCA")

### Phylogenetic PCA - PCA based on GLS-centering and transformed
# projection
# This produces a PCA independent of phylogeny
phylo.tPCA <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy,
GLS = TRUE, transform = TRUE)
summary(phylo.tPCA)
plot(phylo.tPCA, phylo = TRUE, main = "phylo PCA")

### PaCA - Alignment of data to physlogenetic signal rather than axis of
### greatest variation, like in PCA

# OLS method (rotation of PCA)
PaCA.ols <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy,
  align.to.phy = TRUE)
summary(PaCA.ols)
plot(PaCA.ols, phylo = TRUE, main = "PaCA using OLS")

# GLS method (rotation of Phylogenetic PCA)
PaCA.gls <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy,
align.to.phy = TRUE, GLS = TRUE)
summary(PaCA.gls)
plot(PaCA.gls, phylo = TRUE, main = "PaCA using GLS")

# GLS method (rotation of Phylogenetic PCA with transformed data)
PaCA.gls <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy,
align.to.phy = TRUE, GLS = TRUE, transform = TRUE)
summary(PaCA.gls)
plot(PaCA.gls, phylo = TRUE,
  main = "PaCA using GLS and transformed projection")


### Advanced Plotting
gps <- as.factor(c(rep("gp1", 5), rep("gp2", 4))) # Two random groups
par(mar=c(2, 2, 2, 2))
plot(PaCA.ols, pch=22, cex = 1.5, bg = gps, phylo = TRUE)
# Modify options as desired
#  Add things as desired using standard R plotting
text(par()$usr[1], 0.1*par()$usr[3], labels = "PC1 - 45.64%",
  pos = 4, font = 2)
text(0, 0.95*par()$usr[4], labels = "PC2 - 18.80%", pos = 4, font = 2)
```

```
legend("topleft", pch=22, pt.bg = unique(gps), legend = levels(gps))

### 3D plot with a phylogeny and time on the z-axis
plot(PCA.w.phylo, time.plot = TRUE)
plot(PCA.w.phylo, time.plot = TRUE, bg = "red",
   phylo.par = list(tip.labels = TRUE,
tip.txt.cex = 2, edge.color = "blue", edge.width = 2))
```

---

gpagen *Generalized Procrustes analysis of points, curves, and surfaces*

Description

A general function to perform Procrustes analysis of two- or three-dimensional landmark data that can include both fixed landmarks and sliding semilandmarks

Usage

```
gpagen(
  A,
  curves = NULL,
  surfaces = NULL,
  PrinAxes = TRUE,
  max.iter = NULL,
  tol = 1e-04,
  ProcD = FALSE,
  approxBE = FALSE,
  sen = 0.5,
  Proj = TRUE,
  verbose = FALSE,
  print.progress = TRUE,
  Parallel = FALSE
)
```

Arguments

A	Either an object of class geomorphShapes or a 3D array (p x k x n) containing landmark coordinates for a set of specimens. If A is a geomorphShapes object, the curves argument is not needed.
curves	An optional matrix defining which landmarks should be treated as semilandmarks on boundary curves, and which landmarks specify the tangent directions for their sliding. This matrix is generated with `readland.shapes` following digitizing of curves in StereoMorph, or may be generated using the function `define.sliders`.
surfaces	An optional vector defining which landmarks should be treated as semilandmarks on surfaces

PrinAxes	A logical value indicating whether or not to align the shape data by principal axes
max.iter	The maximum number of GPA iterations to perform before superimposition is halted. The final number of iterations will be larger than max.iter, if curves or surface semilandmarks are involved, as max.iter will pertain only to iterations with sliding of landmarks.
tol	A numeric value that can be adjusted for evaluation of the convergence criterion. If the criterion is lower than the tolerance, iterations will be stopped.
ProcD	A logical value indicating whether or not Procrustes distance should be used as the criterion for optimizing the positions of semilandmarks (if not, bending energy is used)
approxBE	A logical value for whether bending energy should be estimated via approximate thin-plate spline (TPS) mapping for sliding semilandmarks. Approximate TPS mapping is much faster and allows for more iterations, which might return more reliable results than few iterations with full TPS. If using full TPS, one should probably change max.iter to be few for large data sets.
sen	A numeric value between 0.1 and 1 to adjust the sensitivity of true bending energy to use in an approximated thin plate mapping of bending energy. This value is the proportion of landmarks (excluding semilandmarks) to seek for estimating bending energy. Sliding semilandmarks are always included in the final set of landmarks used to make estimates, so the actual sensitivity is higher than the chosen value.
Proj	A logical value indicating whether or not the Procrustes aligned specimens should be projected into tangent space
verbose	A logical value for whether to include statistics that take a long time to calculate with large data sets, including a Procrustes distance matrix among specimens, a variance-covariance matrix among Procrustes coordinates (shape variables), and variances of landmark points. Formatting a data frame for coordinates and centroid size is also embedded within this option. This argument should be FALSE unless explicitly needed. For large data sets, it will slow down the analysis, extensively.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen.
Parallel	For sliding semi-landmarks only, either a logical value to indicate whether parallel processing should be used or a numeric value to indicate the number of cores to use in parallel processing via the `parallel` library. If TRUE, this argument invokes forking of all processor cores, except one. If FALSE, only one core is used. A numeric value directs the number of cores to use, but one core will always be spared. Parallel processing is probably only valuable with large data sets.

Details

The function performs a Generalized Procrustes Analysis (GPA) on two-dimensional or three-dimensional landmark coordinates. The analysis can be performed on fixed landmark points, semilandmarks on curves, semilandmarks on surfaces, or any combination. If data are provided in the form of a 3D array, all landmarks and semilandmarks are contained in this object. If this is the only

component provided, the function will treat all points as if they were fixed landmarks. To designate some points as semilandmarks, one uses the "curves=" or "surfaces=" options (or both). To include semilandmarks on curves, a matrix defining which landmarks are to be treated as semilandmarks is provided using the "curves=" option. This matrix contains three columns that specify the semilandmarks and two neighboring landmarks which are used to specify the tangent direction for sliding. The matrix may be generated using the function `define.sliders`). Likewise, to include semilandmarks on surfaces, one must specify a vector listing which landmarks are to be treated as surface semilandmarks using the "surfaces=" option. The "ProcD=FALSE" option (the default) will slide the semilandmarks based on minimizing bending energy, while "ProcD=TRUE" will slide the semilandmarks along their tangent directions using the Procrustes distance criterion. The Procrustes aligned specimens may be projected into tangent space using the "Proj=TRUE" option. The function also outputs a matrix of pairwise Procrustes Distances, which correspond to Euclidean distances between specimens in tangent space if "Proj=TRUE", or to the geodesic distances in shape space if "Proj=FALSE". NOTE: Large datasets may exceed the memory limitations of R.

Generalized Procrustes Analysis (GPA: Gower 1975, Rohlf and Slice 1990) is the primary means by which shape variables are obtained from landmark data (for a general overview of geometric morphometrics see Bookstein 1991, Rohlf and Marcus 1993, Adams et al. 2004, Zelditch et al. 2012, Mitteroecker and Gunz 2009, Adams et al. 2013). GPA translates all specimens to the origin, scales them to unit-centroid size, and optimally rotates them (using a least-squares criterion) until the coordinates of corresponding points align as closely as possible. The resulting aligned Procrustes coordinates represent the shape of each specimen, and are found in a curved space related to Kendall's shape space (Kendall 1984). Typically, these are projected into a linear tangent space yielding Kendall's tangent space coordinates (i.e., Procrustes shape variables), which are used for subsequent multivariate analyses (Dryden and Mardia 1993, Rohlf 1999). Additionally, any semilandmarks on curves and surfaces are slid along their tangent directions or tangent planes during the superimposition (see Bookstein 1997; Gunz et al. 2005). Presently, two implementations are possible: 1) the locations of semilandmarks can be optimized by minimizing the bending energy between the reference and target specimen (Bookstein 1997), or by minimizing the Procrustes distance between the two (Rohlf 2010). Note that specimens are NOT automatically reflected to improve the GPA-alignment.

The generic functions, `print`, `summary`, and `plot` all work with `gpagen`. The generic function, `plot`, calls `plotAllSpecimens`.

Notes for geomorph 4.0:

Starting with geomorph version 4.0, it is possible to use approximated thin-plate spline (TPS) mapping to estimate bending energy (when bending energy is used for sliding semi-landmarks). Approximated TPS has some computational benefit for large data sets (more GPA iterations are possible in a shorter time), but one should make sure first that results are reasonable. Approximated TPS uses a subset of points (semilandmarks) to estimate bending energy locally, where points are free to slide. Some subsets might be misrepresentative of the global (full configuration) bending energy, and strange results are possible. A comparison between TPS and approximated TPS outcomes could be tried with a few specimens to verify consistency of results before using approximated TPS on a large data set.

Choosing to use approximated TPS with only a few sliders is dangerous, as the subset of points might be too small to be reliable. Approximated TPS will work best for data sets with a sufficient number of surface landmarks. If a data set is nearly 100

Notes for geomorph 3.0:

Compared to older versions of geomorph, users might notice subtle differences in Procrustes shape variables when using semilandmarks (curves or surfaces). This difference is a result of using recursive updates of the consensus configuration with the sliding algorithms (minimized bending energy or Procrustes distances). (Previous versions used a single consensus through the sliding algorithms.) Shape differences using the recursive updates of the consensus configuration should be highly correlated with shape differences using a single consensus during the sliding algorithm, but rotational "flutter" can be expected. This should have no qualitative effect on inferential analyses using Procrustes residuals.

Value

An object of class gpagen returns a list with the following components:

`coords`	A (p x k x n) array of Procrustes shape variables, where p is the number of landmark points, k is the number of landmark dimensions (2 or 3), and n is the number of specimens. The third dimension of this array contains names for each specimen if specified in the original input array.
`Csize`	A vector of centroid sizes for each specimen, containing the names for each specimen if specified in the original input array.
`iter`	The number of GPA iterations until convergence was found (or GPA halted).
`points.VCV`	Variance-covariance matrix among Procrustes shape variables.
`points.var`	Variances of landmark points.
`consensus`	The consensus (mean) configuration.
`procD`	Procrustes distance matrix for all specimens (see details). Note that for large data sets, R might return a memory allocation error, in which case the error will be suppressed and this component will be NULL. For such cases, users can augment memory allocation and create distances with the dist function, independent from gpagen, using the coords or data output.
`p`	Number of landmarks.
`k`	Number of landmark dimensions.
`nsliders`	Number of semilandmarks along curves.
`nsurf`	Number of semilandmarks as surface points.
`data`	Data frame with an n x (pk) matrix of Procrustes shape variables and centroid size.
`Q`	Final convergence criterion value.
`slide.method`	Method used to slide semilandmarks.
`call`	The match call.

Author(s)

Dean Adams and Michael Collyer

References

Adams, D. C., F. J. Rohlf, and D. E. Slice. 2004. Geometric morphometrics: ten years of progress following the 'revolution'. It. J. Zool. 71:5-16.

Adams, D. C., F. J. Rohlf, and D. E. Slice. 2013. A field comes of age: Geometric morphometrics in the 21st century. Hystrix.24:7-14.

Bookstein, F. L. 1991. Morphometric tools for landmark data: Geometry and Biology. Cambridge Univ. Press, New York.

Bookstein, F. L. 1997. Landmark methods for forms without landmarks: morphometrics of group differences in outline shape. 1:225-243.

Dryden, I. L., and K. V. Mardia. 1993. Multivariate shape analysis. Sankhya 55:460-480.

Gower, J. C. 1975. Generalized Procrustes analysis. Psychometrika 40:33-51.

Gunz, P., P. Mitteroecker, and F. L. Bookstein. 2005. semilandmarks in three dimensions. Pp. 73-98 in D. E. Slice, ed. Modern morphometrics in physical anthropology. Klewer Academic/Plenum, New York.

Kendall, D. G. 1984. Shape-manifolds, Procrustean metrics and complex projective spaces. Bulletin of the London Mathematical Society 16:81-121.

Mitteroecker, P., and P. Gunz. 2009. Advances in geometric morphometrics. Evol. Biol. 36:235-247.

Rohlf, F. J., and D. E. Slice. 1990. Extensions of the Procrustes method for the optimal superimposition of landmarks. Syst. Zool. 39:40-59.

Rohlf, F. J., and L. F. Marcus. 1993. A revolution in morphometrics. Trends Ecol. Evol. 8:129-132.

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

Rohlf, F. J. 2010. tpsRelw: Relative warps analysis. Version 1.49. Department of Ecology and Evolution, State University of New York at Stony Brook, Stony Brook, NY.

Zelditch, M. L., D. L. Swiderski, H. D. Sheets, and W. L. Fink. 2012. Geometric morphometrics for biologists: a primer. 2nd edition. Elsevier/Academic Press, Amsterdam.

Examples

```
# Example 1: fixed points only
data(plethodon)
Y.gpa <- gpagen(plethodon$land, PrinAxes = FALSE)
summary(Y.gpa)
plot(Y.gpa)

# Example 2: points and semilandmarks on curves
data(hummingbirds)

###Slider matrix
hummingbirds$curvepts

# Using bending energy for sliding
Y.gpa <- gpagen(hummingbirds$land, curves = hummingbirds$curvepts,
ProcD = FALSE)
```

```
summary(Y.gpa)
plot(Y.gpa)


# Using Procrustes Distance for sliding
Y.gpa <- gpagen(hummingbirds$land, curves = hummingbirds$curvepts,
ProcD = TRUE)
summary(Y.gpa)
plot(Y.gpa)


# Example 3: points, curves and surfaces
data(scallops)

# Using Procrustes Distance for sliding
Y.gpa <- gpagen(A = scallops$coorddata, curves = scallops$curvslide,
surfaces = scallops$surfslide)
# NOTE can summarize as: summary(Y.gpa)
# NOTE can plot as: plot(Y.gpa)
```

---

gridPar *Set up parameters for grids, points, and links in plotRefToTarget*

Description

Function produces a list of parameter changes within plotRefToTarget.

Usage

```
gridPar(
  pt.bg = "gray",
  pt.size = 1.5,
  link.col = "gray",
  link.lwd = 2,
  link.lty = 1,
  out.col = "gray",
  out.cex = 0.1,
  tar.pt.bg = "black",
  tar.pt.size = 1,
  tar.link.col = "black",
  tar.link.lwd = 2,
  tar.link.lty = 1,
  tar.out.col = "black",
  tar.out.cex = 0.1,
  n.col.cell = 20,
  grid.col = "black",
  grid.lwd = 1,
  grid.lty = 1,
```

```
  txt.adj = NULL,
  txt.pos = 2,
  txt.cex = 0.8,
  txt.col = "black"
)
```

Arguments

pt.bg	Background color of reference configuration points (single value or vector of values)
pt.size	Scale factor for reference configuration points (single value or vector of values)
link.col	The color of links for reference configurations (single value or vector of values)
link.lwd	The line weight of links for reference configurations (single value or vector of values)
link.lty	The line type of links for reference configurations (single value or vector of values)
out.col	The color of outline for reference configurations (single value or vector of values)
out.cex	The size of plotting symbol of outline for reference configurations (single value or vector of values)
tar.pt.bg	Background color of target configuration points (single value or vector of values)
tar.pt.size	Scale factor for target configuration points (single value or vector of values)
tar.link.col	The color of links for target configurations (single value or vector of values)
tar.link.lwd	The line weight of links for target configurations (single value or vector of values)
tar.link.lty	The line type of links for target configurations (single value or vector of values)
tar.out.col	The color of outline for target configurations (single value or vector of values)
tar.out.cex	The size of plotting symbol of outline for target configurations (single value or vector of values)
n.col.cell	The number of square cells (along x axis) for grids (single numerical value)
grid.col	The color of grid lines (single value)
grid.lwd	Scale factor for the weight of grid lines (single numerical value)
grid.lty	The line type for grid lines (single numerical value, as in base R plot)
txt.adj	The adjustment value of the landmark label (one or two values, as in base R text and text3d)
txt.pos	The position of the landmark label, overrides txt.adj (single numerical value, as in base R text and text3d)
txt.cex	The size of the landmark label text (single numerical value, as in base R text and text3d)
txt.col	The color of the landmark label text (single numerical value, as in base R text and text3d)

Details

The function allows users to vary certain plotting parameters to produce different graphical outcomes for plotRefToTarget. Not all parameters need to be adjusted to use this function, as the defaults above will be used.

Author(s)

Michael Collyer & Emma Sherratt

See Also

plotRefToTarget, text, text3d

Examples

```
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
ref<-mshape(Y.gpa$coords)
plotRefToTarget(ref,Y.gpa$coords[,,39]) # default settings

# Altering points and links
GP1 <- gridPar(pt.bg = "red", pt.size = 1, link.col="blue", link.lwd=2,
n.col.cell=50)
plotRefToTarget(ref,Y.gpa$coords[,,39], gridPars=GP1, mag=2,
links=plethodon$links, method="TPS")

# Altering point color
GP2 <- gridPar(pt.bg = "green", pt.size = 1)
plotRefToTarget(ref,Y.gpa$coords[,,39], gridPars=GP2, mag=3,
method="vector")

# Altering ref and target points
GP3 <- gridPar(pt.bg = "blue", pt.size = 1.5, tar.pt.bg = "orange",
tar.pt.size = 1)
plotRefToTarget(ref,Y.gpa$coords[,,39], gridPars=GP3, mag=3,
method="points")

# Altering outline color
GP4 <- gridPar(tar.out.col = "red", tar.out.cex = 0.3)
plotRefToTarget(ref,Y.gpa$coords[,,39], gridPars=GP4, mag=3,
outline=plethodon$outline, method="TPS")

# Altering text labels
GP5 <- gridPar(txt.pos = 3, txt.col = "red")
plotRefToTarget(ref,Y.gpa$coords[,,39], gridPars=GP5, mag=3,
method="vector", label=TRUE)
```

hummingbirds *Landmark data from hummingbird bills (includes sliding semilandmarks on curves)*

Description

Landmark data from hummingbird bills (includes sliding semilandmarks on curves)

Author(s)

Chelsea Berns and Dean Adams

References

Berns, C.M., and Adams, D.C. 2010. Bill shape and sexual shape dimorphism between two species of temperate hummingbirds: Archilochus alexandri (black-chinned hummingbirds) and Archilochus colubris (ruby-throated hummingbirds). The Auk. 127:626-635.

integration.test *Quantify morphological integration between modules*

Description

Function quantifies the degree of morphological integration between modules of Procrustes shape variables

Usage

```
integration.test(
  A,
  A2 = NULL,
  partition.gp = NULL,
  iter = 999,
  seed = NULL,
  print.progress = TRUE
)
```

Arguments

A	A 3D array (p x k x n) containing Procrustes shape variables for all specimens, or a matrix (n x variables)
A2	An optional 3D array (p x k x n) containing Procrustes shape variables for all specimens, or a matrix (n x variables) for a second partition
partition.gp	A list of which landmarks (or variables) belong in which partition: (e.g. A, A, A, B, B, B, C, C, C). Required when only 1 dataset provided.

iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function quantifies the degree of morphological integration between modular partitions of shape data as defined by Procrustes shape variables. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. The function may be used to assess the degree of morphological integration between two or more sets of variables.

The function estimates the degree of morphological integration using a two-block partial least squares analysis (PLS). When used with landmark data, this analysis is referred to as singular warps analysis (Bookstein et al. 2003). If more than two partitions are defined, the average pairwise PLS correlation is utilized as the test statistic. The observed test value is then compared to a distribution of values obtained by randomly permuting the individuals (rows) in one partition relative to those in the other. A significant result is found when the observed PLS correlation is large relative to this distribution, and implies that the structures are integrated with one another (see Bookstein et al. 2003). In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019). If only two partitions are specified, a plot of PLS scores along the first set of PLS axes is optionally displayed, and thin-plate spline deformation grids along these axes are also shown if data were input as a 3D array.

Input for the analysis can take one of two forms. First, one can input a single dataset (as a matrix or 3D array, along with a vector describing which variables correspond to which partitions (for the case of a 3D array, which landmarks belong to which partitions is specified). Alternatively, when evaluating the integration between two structures or partitions, two datasets may be provided.

The generic functions, print, summary, and plot all work with integration.test. The generic function, plot, produces a two-block.pls plot. This function calls plot.pls, which produces an ordination plot. An additional argument allows one to include a vector to label points. Starting with version 3.1.0, warpgrids are no longer available with plot.pls but after making a plot, the function returns values that can be used with picknplot.shape or a combination of shape.predictor and plotRefToTarget to visualize shape changes in the plot (via warpgrids).

Similarity to two.b.pls and compare.pls :

Note that integration.test performed on two matrices or arrays returns the same results as two.b.pls. However, two.b.pls is limited to only two modules. It might be of interest with 3+ modules to perform integration tests between all pairwise comparisons of modules. This can be done, test by test, and the levels of integration can be compared with compare.pls. Such results are different than using the average amount of integration, as performed by integration.test when more than two modules are input.

Using phylogenies and PGLS:

If one wishes to incorporate a phylogeny, `phylo.integration` is the function to use. This function is exactly the same as `integration.test` but allows PGLS estimation of PLS vectors. Because `integration.test` can be used on two blocks, `phylo.integration` likewise allows one to perform a phylogenetic two-block PLS analysis.

Notes for geomorph 3.0.4 and subsequent versions:

Compared to previous versions of geomorph, users might notice differences in effect sizes. Previous versions used z-scores calculated with expected values of statistics from null hypotheses (sensu Collyer et al. 2015); however Adams and Collyer (2016) showed that expected values for some statistics can vary with sample size and variable number, and recommended finding the expected value, empirically, as the mean from the set of random outcomes. Geomorph 3.0.4 and subsequent versions now center z-scores on their empirically estimated expected values and where appropriate, log-transform values to assure statistics are normally distributed. This can result in negative effect sizes, when statistics are smaller than expected compared to the average random outcome. For ANOVA-based functions, the option to choose among different statistics to measure effect size is now a function argument.

Value

Objects of class "pls" from integration.test return a list of the following:

`r.pls`	The estimate of morphological integration: PLS.corr. The mean of pairwise PLS correlations between partitions is used when there are more than two partitions.
`r.pls.mat`	The pairwise r.pls, if the number of partitions is greater than 2.
`P.value`	The empirically calculated P-value from the resampling procedure.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`left.pls.vectors`	The singular vectors of the left (x) block (for 2 modules only).
`right.pls.vectors`	The singular vectors of the right (y) block (for 2 modules only).
`random.r`	The correlation coefficients found in each random permutation of the resampling procedure.
`XScores`	Values of left (x) block projected onto singular vectors (for 2 modules only).
`YScores`	Values of right (y) block projected onto singular vectors (for 2 modules only).
`svd`	The singular value decomposition of the cross-covariances (for 2 modules only).
`A1`	Input values for the left block (for 2 modules only).
`A2`	Input values for the right block (for 2 modules only).
`A1.matrix`	Left block (matrix) found from A1 (for 2 modules only).
`A2.matrix`	Right block (matrix) found from A2 (for 2 modules only).
`permutations`	The number of random permutations used in the resampling procedure.
`call`	The match call.

Author(s)

Dean Adams

References

Bookstein, F. L., P. Gunz, P. Mitteroecker, H. Prossinger, K. Schaefer, and H. Seidler. 2003. Cranial integration in Homo: singular warps analysis of the midsagittal plane in ontogeny and evolution. J. Hum. Evol. 44:167-187.

Collyer, M.L., D.J. Sekora, and D.C. Adams. 2015. A method for analysis of phenotypic change for phenotypes described by high-dimensional data. Heredity. 115:357-365.

Adams, D.C. and M.L. Collyer. 2016. On the comparison of the strength of morphological integration across morphometric datasets. Evolution. 70:2623-2631.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

See Also

two.b.pls, modularity.test, phylo.integration, and compare.pls

Examples

```
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
#landmarks on the skull and mandible assigned to partitions
land.gps<-c("A","A","A","A","A","B","B","B","B","B","B","B")
IT <- integration.test(Y.gpa$coords, partition.gp=land.gps, iter=999)
summary(IT) # Test summary
plot(IT) # PLS plot


 ### Visualize shape variation using picknplot.shape Because picknplot
 ### requires user decisions, the following example
 ### is not run (but can be with removal of #).
 ### For detailed options, see the picknplot help file
 # picknplot.shape(plot(IT))

IT$left.pls.vectors # extracting just the left (first block)
# singular vectors
```

`interlmkdist` *Calculate linear distances between landmarks*

Description

A function to calculate linear distances between a set of landmark coordinates (interlandmark distances)

Usage

```
interlmkdist(A, lmks)
```

Arguments

A	A 3D array (p x k x n) containing landmark coordinates for a set of specimens
lmks	A matrix or dataframe of landmark addresses for the start and end landmarks defining m linear measurements (can be either 2-x-m or m-x-2). Either the rows or the columns should have names 'start' and 'end' to define landmarks.

Details

Function takes a 3D array of landmark coordinates from a set of specimens and the addresses for the start and end landmarks defining linear measurements and then calculates the interlandmark distances. The function returns a matrix of linear distances for all specimens. If the 'lmks' matrix has row or column names defining the name of the linear measurements, the returned matrix will use these for column names (see example). If only two interlandmark distances, 'lmks' input must be m x 2.

Value

Function returns a matrix (n x m) of m linear distances for n specimens

Author(s)

Emma Sherratt & Michael Collyer

Examples

```
data(plethodon)
# Make a matrix defining three interlandmark distances
lmks <- matrix(c(8,9,6,12,4,2), ncol=2, byrow=TRUE,
dimnames = list(c("eyeW", "headL", "mouthL"),c("start", "end")))
# where 8-9 is eye width; 6-12 is head length; 4-2 is mouth length
# or alternatively
lmks <- data.frame(eyeW = c(8,9), headL = c(6,12), mouthL = c(4,2),
row.names = c("start", "end"))
A <- plethodon$land
lineardists <- interlmkdist(A, lmks)
```

---

larvalMorph *Head and tail shapes of larval salamanders*

Description

Landmark data from heads and tails of larval Salamanders exposed to different treatments of herbicides.

Details

Data set includes tail landmarks (coords), index for identifying semilandmarks (sliders), and vectors for variables including herbicide treatment (Treatment) and Family (Family). The latter variable indicates the egg masses (clutches) sampled in the wild from which individual eggs were randomly assigned to treatment. See Levis et al. (2016) for more experimental details.

Author(s)

Michael Collyer

References

Levis, N.A, M.L. Schooler, J.R. Johnson, and M.L. Collyer. 2016. The effects of terrestrial and aquatic herbicides on larval salamander morphology and swim speed. Biological Journal of the Linnean Society. 118:569-581.

---

lizards *Dorsal head shape data of lizards*

Description

Superimposed landmark data of the dorsal view of lizard heads

Details

Dataset includes superimposed landmarks (coords), centroid size (cs), an index of individuals (ind) and digitizing repetitions (rep), and a table of symmetrical matching landmarks (lm.pairs). The object is a `geomorph.data.frame`. The dataset corresponds to the data for population "b" from Lazic et al. 2015.

Author(s)

Antigoni Kaliontzopoulou

References

Lazic, M., Carretero, M.A., Crnobrnja-Isailovic, J. & Kaliontzopoulou, A. 2015. Effects of environmental disturbance on phenotypic variation: an integrated assessment of canalization, developmental stability, modularity and allometry in lizard head shape. American Naturalist 185: 44-58.

make_ggplot *Convert geomorph plots to ggplot objects*

Description

Function attempts to coerce plot information from a geomorph plot object to an amenable ggplot object.

Usage

```
make_ggplot(object)
```

Arguments

object	A plot object produced from plot.gm.prcomp, plot.pls,plot.procD.lm, or plotAllometry. For plot.procD.lm objects, only types "PC" or "regression" should work.

Details

This function will attempt to use the plot arguments from an geomorph plot object to make a ggplot that can be additionally updated, as desired. Not all plot characteristics might be converted. Nonetheless, a ggplot will be coerced and could be updated, according to user preference.

This function assumes no responsibility for arguments made by ggplot. It merely produces a ggplot object that should resemble a geomorph plot default. Any augmentation of ggplot objects can be done either by direct intervention of the ggplot produced or reformatting the initial geomorph plot produced. One should not expect direct correspondence between R base plot parameters and ggplot parameters.

Author(s)

Michael Collyer

Examples

```
### PLS Example
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
# landmarks on the skull and mandible assigned to partitions
land.gps<-c("A","A","A","A","A","B","B","B","B","B","B","B")
IT <- integration.test(Y.gpa$coords, partition.gp=land.gps, iter=999)
summary(IT) # Test summary
P <- plot(IT) # PLS plot
make_ggplot(P) # same plot in ggplot

### Allometry example
```

```
data(plethodon)
Y.gpa <- gpagen(plethodon$land, print.progress = FALSE)    #GPA-alignment

gdf <- geomorph.data.frame(Y.gpa, site = plethodon$site,
                           species = plethodon$species)

fit <- procD.lm(coords ~ Csize * species * site, data=gdf, iter=0,
                print.progress = FALSE)

P <- plotAllometry(fit, size = gdf$Csize, logsz = TRUE, method = "PredLine",
                   pch = 19, col = as.numeric(interaction(gdf$species, gdf$site)))

make_ggplot(P)

### Tangent Space plot

data(plethspecies)
Y.gpa <- gpagen(plethspecies$land)    #GPA-alignment

PCA.w.phylo <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy)
P <- plot(PCA.w.phylo, phylo = TRUE, main = "PCA.w.phylo")
make_ggplot(P)
```

`modularity.test` *Evaluate the degree of modular signal in shape data*

Description

Function quantifies the degree of modularity between two or more hypothesized modules of Procrustes shape variables and compares this to patterns found by randomly assigning landmarks into subsets

Usage

```
modularity.test(
  A,
  partition.gp,
  iter = 999,
  CI = FALSE,
  seed = NULL,
  opt.rot = TRUE,
  print.progress = TRUE
)
```

Arguments

`A`	A 3D array (p x k x n) containing Procrustes shape variables for all specimens, or a matrix (n x variables)

partition.gp	A list of which landmarks (or variables) belong in which partition: (e.g. A, A, A, B, B, B, C, C, C)
iter	Number of iterations for significance testing
CI	A logical argument indicating whether bootstrapping should be used for estimating confidence intervals
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
opt.rot	A logical argument for whether the optimal rotation for CR should be used for landmark data (default = TRUE)
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function quantifies the degree of modularity in two or more hypothesized modules of Procrustes shape variables, and compares this to what is expected under the null hypothesis of random assignment of variables to partitions (i.e., neither modular nor integrated structure). Input may be either a 2D matrix of phenotypic values, or a 3D array of Procrustes shape variables. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA); e.g., with gpagen. The degree of modularity is quantified using the CR coefficient (Adams 2016). If more than two modules are defined, the average pairwise CR coefficient is utilized. The CR coefficient for the observed modular hypothesis is then compared to a distribution of values obtained by randomly assigning landmarks into subsets, with the restriction that the number of landmarks in each subset is identical to that observed in each of the original partitions. A significant modular signal is found when the observed CR coefficient is small relative to this distribution (see Adams 2016). Such a result implies that there is significantly greater independence among modules than is expected under the null hypothesis of random associations of variables (neither modular nor integrated structure). This result is consistent with the identification of significant modular structure in the data. In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019). A histogram of coefficients obtained via resampling is presented, with the observed value designated by an arrow in the plot. For landmark data, the CR coefficient found from the average CR across a 90 degree rotation of the data is used as the test statistic (see Adams 2016). For all data, the CR coefficient is returned, and (optionally) its 95 for landmark data, estimation of the CI can take some time to compute.

Landmark groups can be defined using define.modules, or made by hand (see example below). To use this method with other data (i.e., a set of length measurements), the input A should be a matrix of n rows of specimens and variables arranged in columns. In this case, the partition.gp input should have each variable assigned to a partition.

The generic functions, print, summary, and plot all work with modularity.test. The generic function, plot, produces a histogram of random CR values associated with the resampling procedure.

Value

An object of class "CR" is a list containing the following

`CR`	Covariance ratio: The estimate of the observed modular signal.
`CInterval`	The bootstrapped 95 percent confidence intervals of the CR, if CI = TRUE.
`CR.boot`	The bootstrapped CR values, if CI = TRUE
`P.value`	The empirically calculated P-value from the resampling procedure.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`CR.mat`	For more than two partitions, the pairwise CRs among partitions.
`random.CR`	The CR calculated in each of the random permutations of the resampling procedure.
`permutations`	The number of random permutations used in the resampling procedure.
`call`	The match call.

Author(s)

Dean Adams

References

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.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

See Also

`two.b.pls`, `integration.test`, `phylo.modularity`, and `phylo.integration`

Examples

```
data(pupfish)
Y.gpa<-gpagen(pupfish$coords, print.progress = FALSE)    #GPA-alignment
 #landmarks on the body and operculum
land.gps<-rep('a',56); land.gps[39:48]<-'b'

MT <- modularity.test(Y.gpa$coords,land.gps,CI=FALSE,iter=49)
## NOTE: for actual analysis one should increase iterations!
summary(MT) # Test summary
plot(MT) # Histogram of CR sampling distribution
# Result implies modularity present
```

`morphol.disparity` *Morphological disparity for one or more groups of specimens*

Description

Function estimates morphological disparity and performs pairwise comparisons among groups

Usage

```
morphol.disparity(
  f1,
  groups = NULL,
  partial = FALSE,
  transform = FALSE,
  iter = 999,
  seed = NULL,
  data = NULL,
  print.progress = TRUE,
  ...
)
```

Arguments

`f1`	A formula describing the linear model used. The left-hand portion of the formula should be a 3D array (p x k x n) containing Procrustes shape variables for a set of specimens, or a matrix (n x variables). The right-hand portion of the formula should be " ~1" to use the overall mean, or "~ x1 + x2 + x3 +...", where each x is a covariate or factor. (Interactions and nested terms also work.) Alternatively, one can use an object of class "procD.lm" or "lm.rrpp", which already has a formula defined. This is especially helpful for analyses performed with `procD.pgls`, as residuals from PGLS will be used for analysis (see examples).
`groups`	Either a formula designating groups, e.g., groups = ~ groups, or a factor, or a vector coercible to factor. If NULL and if f1 is a procD.lm or lm.rrpp model fit, morphol.disparity will attempt to define groups based on the terms of the model. If there are no groups inherently indicated in f1 and groups is NULL, a single Procrustes variance will be returned for the entire data set.
`partial`	A logical value to indicate whether partial disparities should be calculated, sensu Foote (1993). If TRUE, the model formula should have only an intercept (e.g., coords ~ 1); otherwise an error will be returned.
`transform`	A logical value to indicate whether disparity should be measured in the transformed space rather than the tangent space. This is only applicable if a PGLS model is used, in which case the transformation would be necessary for a variance that is independent of phylogeny. If transform = FALSE, disparity tracks the dispersion one would observe in a phylo-PC plot. If transform = TRUE, disparity tracks the dispersion in a PC plot of the transformed space (lacking phylogenetic signal). See Collyer and Adams 2021, and examples below.

iter	Number of iterations for permutation test
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
data	A data frame for the function environment, see geomorph.data.frame
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.
...	Arguments passed on to procD.lm (typically associated with the lm function, such as weights or offset, plus SS.type).

Details

The function estimates morphological disparity and performs pairwise comparisons to identify differences among groups. Morphological disparity is estimated as the Procrustes variance, overall or for groups, using residuals of a linear model fit. Procrustes variance is the same sum of the diagonal elements of the group covariance matrix divided by the number of observations in the group (e.g., Zelditch et al. 2012). The function takes as input a formula to describe the linear model fit, plus a formulaic indication of groups (e.g., ~ groups). It is assumed that the formula describes shape data that have been GPA-aligned [e.g., gpagen], although the function can work with any multivariate data.

Partial disparities (Foote 1993) can also be calculated, but only for model formulas containing only an intercept (e.g., coords ~ 1). Partial disparity has the same numerator as Procrustes variance (with respect to an overall mean) but the denominator is N - 1 for all N observations, rather than n, the group size. (The sum of all group n equals N.) Partial disparities have the appeal that the sum of group partial disparities it the total disparity.

Absolute differences in Procrustes variances are test statistics that can be used to test differences in morphological disparity among groups. These differences are statistically evaluated through permutation, where the vectors of residuals are randomized among groups. The function can be used to obtain disparity for the whole dataset by using "a dummy group factor "~ 1" as the right-hand portion of the formula, in which case only Procrustes variance is returned. Additionally, if the right-hand portion of the formula only contains (continuous) covariates, e.g., "~ Csize", Procrustes variance will be calculated for the whole data set or groups, after accounting for the linear regression described. Finally, different factors can be indicated in the formula and for groups, if one wishes to compare morphological disparities for groups comprising only a portion of or collapsing of the groups in a more complex model (see examples).

This function can be used with an object of class "procD.lm" or "lm.rrpp", if such analyses have already been performed. This is especially useful for analyses performed with procD.pgls. In this case, residuals obtained from PGLS estimation of coefficients, rather than OLS estimation, will be used in the analysis. Thus, one can account for phylogeny when comparing morphological disparity among groups. However, one should be aware that PGLS finds the PGLS residuals and transforms them via the phylogenetic covariance matrix in order to calculate variance. Thus, disparity (dispersion in tangent space) and variance (dispersion in transformed space, after accounting for phylogeny) can be considered different things. This function uses an argument, transform, to allow users to use either GLS residuals in tangent space or transformed residuals in the transformed

space to calculate disparity. The former characterizes the dispersion of actual shapes around GLS means; the latter estimates GLS variances, by group.

Notes for geomorph 3.1.0 and subsequent versions:

The function `pairwise` in the RRPP package can also be used to evaluate morphological disparity, and will yield results identical to those of the current function. A simple example is shown below

Value

Objects of class "morphol.disparity" return a list with the following components (if groups are specified):

`Procrustes.var`	Observed Procrustes variances.
`PV.dist`	Observed pairwise absolute differences (as distances) among group Procrustes variances.
`PV.dist.Pval`	P-values associated with pairwise differences.
`random.PV.dist`	Pairwise distance matrices produced in the resampling procedure.
`permutations`	Number of random permutations in resampling procedure.
`partial`	Logical value to indicate if partial disparities were calculated.
`call`	The match call

Author(s)

Emma Sherratt and Michael Collyer

References

Zelditch, M. L., D. L. Swiderski, H. D. Sheets, and W. L. Fink. 2012. Geometric morphometrics for biologists: a primer. 2nd edition. Elsevier/Academic Press, Amsterdam.

Foote, M. 1993. Contributions of individual taxa to overall morphological disparity. Paleobiology, 19: 403-419.

Collyer, M.L and D.C. Adams, 2021. Phylogenetically Aligned component analysis. Methods in Ecology and Evolution, 12: 369-372.

Examples

```
data(plethodon)
Y.gpa<-gpagen(plethodon$land, print.progress = FALSE)    #GPA-alignment
gdf <- geomorph.data.frame(Y.gpa, species = plethodon$species,
site = plethodon$site)

# Morphological disparity for entire data set
morphol.disparity(coords ~ 1, groups = NULL, data = gdf,
iter = 999, print.progress = FALSE)

# Morphological disparity for entire data set, accounting for allometry
morphol.disparity(coords ~ Csize, groups= NULL, data = gdf,
iter = 999, print.progress = FALSE)
```

```
# Morphological disparity without covariates, using overall mean
morphol.disparity(coords ~ 1, groups= ~ species*site, data = gdf,
iter = 999, print.progress = FALSE)

# Morphological partial disparities for overal mean
morphol.disparity(coords ~ 1, groups= ~ species*site, partial = TRUE,
data = gdf, iter = 999, print.progress = FALSE)

# Morphological disparity without covariates, using group means
morphol.disparity(coords ~ species*site, groups= ~species*site,
data = gdf, iter = 999, print.progress = FALSE)

# Morphological disparity of different groups than those
# described by the linear model
morphol.disparity(coords ~ Csize + species*site, groups= ~ species,
data = gdf, iter = 999, print.progress = FALSE)

# Extracting components
MD <- morphol.disparity(coords ~ Csize + species*site, groups= ~ species,
data = gdf, iter = 999, print.progress = FALSE)
MD$Procrustes.var # just the Procrustes variances


### Morphol.disparity can be used with previously-defined
### procD.lm or lm.rrpp class objects

data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment
gp.end<-factor(c(0,0,1,0,0,1,1,0,0))  #endangered species vs. rest
names(gp.end)<-plethspecies$phy$tip

gdf <- geomorph.data.frame(Y.gpa, phy = plethspecies$phy,
gp.end = gp.end)

pleth.ols <- procD.lm(coords ~ Csize + gp.end,
data = gdf, iter = 999) # ordinary least squares
pleth.pgls <- procD.pgls(coords ~ Csize + gp.end, phy = phy,
data = gdf, iter = 999) # phylogenetic generalized least squares

summary(pleth.ols)
summary(pleth.pgls)

morphol.disparity(f1 = pleth.ols, groups = ~ gp.end, data = gdf,
iter = 999, print.progress = FALSE)
morphol.disparity(f1 = pleth.pgls, groups = ~ gp.end,
transform = FALSE, data = gdf,
iter = 999, print.progress = FALSE) # disparity in tangent space
morphol.disparity(f1 = pleth.pgls, groups = ~ gp.end,
transform = TRUE, data = gdf,
iter = 999, print.progress = FALSE) # disparity in transformed space

# Three plots that correspond to the three tests
```

```
PCA <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy)
pPCA <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy,
GLS = TRUE, transform = FALSE)
tpPCA <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy,
GLS = TRUE, transform = TRUE)

par(mfrow = c(1,3))

# Phylomorphospace
PC.plot <- plot(PCA, pch = 19, phylo = TRUE, main = "PCA-OLS")
shapeHulls(PC.plot, groups = gp.end)

# Phylo-PCA
pPC.plot <- plot(pPCA, pch = 19, phylo = TRUE, main = "pPCA - GLS, not transformed")
shapeHulls(pPC.plot, groups = gp.end)

# Transformed phylo-PCA
tpPC.plot <- plot(tpPCA, pch = 19, phylo = TRUE, main = "tpPCA - GLS, transformed")
shapeHulls(tpPC.plot, groups = gp.end)

par(mfrow = c(1,1))

 ### Variance using RRPP (not necessarily the same as disparity)
 PW <- pairwise(pleth.ols, groups = gp.end)
 summary(PW, test.type = 'var')
 PW2 <- pairwise(pleth.pgls, groups = gp.end)
 summary(PW2, test.type = 'var')
```

---

mosquito *Landmarks on mosquito wings*

Description

Landmarks on mosquito wings

Author(s)

Dean Adams

---

mshape *Estimate mean shape for a set of aligned specimens*

Description

Estimate the mean shape for a set of aligned specimens

Usage

```
mshape(A, na.action = 1)
```

Arguments

A	Either a list (length n, p x k), A 3D array (p x k x n), or a matrix (n x pk) containing GPA-aligned coordinates for a set of specimens
na.action	An index for how to treat missing values: 1 = stop analysis; 2 = return NA for coordinates with missing values for any specimen; 3 = attempt to calculate means for coordinates for all non-missing values.

Details

The function estimates the average landmark coordinates for a set of aligned specimens. Three different methods are available for missing data (see Arguments and Examples).

One can then use the generic function plot to produce a numbered plot of landmark positions and potentially add links, in order to review landmark positions

Author(s)

Antigoni Kaliontzopoulou & Michael Collyer

Examples

```
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
A <- Y.gpa$coords
A[[1]] <- NA # make a missing value, just for example

mshape(Y.gpa$coords)   # mean (consensus) configuration
# mshape(A, na.action = 1) # will return an error
mshape(A, na.action = 2) # returns NA in spot of missing value
mshape(A, na.action = 3) # finds mean values from all possible values
```

---

na.omit.geomorph.data.frame

Handle missing values in rrpp.data.frame objects

Description

Handle missing values in rrpp.data.frame objects

Usage

```
## S3 method for class 'geomorph.data.frame'
na.omit(object, ...)
```

Arguments

object	object (from geomorph.data.frame)
...	further arguments (currently not used)

Author(s)

Michael Collyer

Examples

```
data(plethspecies)
Y.gpa <- gpagen(plethspecies$land, verbose = TRUE)
gdf <- geomorph.data.frame(Y.gpa)
gdf$d <- Y.gpa$procD
gdf$group <- c(rep(1, 4), rep(2, 4), NA) # one unknown group designation
gdf
ndf <- na.omit(gdf)
ndf
```

---

phylo.integration *Quantify phylogenetic morphological integration between two or more sets of variables under Brownian motion*

Description

Function quantifies the degree of phylogenetic morphological covariation between two or more sets of Procrustes shape variables using partial least squares.

Usage

```
phylo.integration(
  A,
  A2 = NULL,
  phy,
  partition.gp = NULL,
  iter = 999,
  seed = NULL,
  print.progress = TRUE
)
```

Arguments

A	A 2D array (n x [p1 x k1]) or 3D array (p1 x k1 x n) containing Procrustes shape variables for the first block
A2	An optional 2D array (n x [p2 x k2]) or 3D array (p2 x k2 x n) containing Procrustes shape variables for the second block
phy	A phylogenetic tree of class phylo - see read.tree in library ape

partition.gp	A list of which landmarks (or variables) belong in which partition: (e.g. A, A, A, B, B, B, C, C, C). This is required when only 1 dataset provided.
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function quantifies the degree of phylogenetic morphological integration between two or more sets of Procrustes shape variables. The approach is based on a Brownian motion model of evolution. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen].

The function estimates the degree of morphological covariation between two or sets of variables while accounting for phylogeny using partial least squares (Adams and Felice 2014), and under a Brownian motion model of evolution. If more than two partitions are defined, the average pairwise PLS correlation is utilized as the test statistic. The observed value is statistically assessed using permutation, where data for one partition are permuted relative to the other partitions. In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019). Note that this permutation is performed on phylogenetically- transformed data, so that the probability of phylogenetic association of A vs. B is similar to that of B vs. A: i.e., prob(A,B|phy)~prob(B,A|phy); thus, shuffling the correct exchangeable units under the null hypothesis of no integration (Adams and Collyer 2018).

Input for the analysis can take one of two forms. First, one can input a single dataset (as a matrix or 3D array, along with a vector describing which variables correspond to which partitions (for the case of a 3D array, which landmarks belong to which partitions is specified). Alternatively, when evaluating the integration between two structures or partitions, two datasets may be provided.

The generic functions, print, summary, and plot all work with phylo.integration. The generic function, plot, produces a two-block.pls plot. This function calls plot.pls, which produces an ordination plot. An additional argument allows one to include a vector to label points. Starting with version 3.1.0, warpgrids are no longer available with plot.pls but after making a plot, the function returns values that can be used with picknplot.shape or a combination of shape.predictor and plotRefToTarget to visualize shape changes in the plot (via warpgrids).

Similarity to two.b.pls and compare.pls :

Note that phylo.integration performed on two matrices or arrays returns the same results as a phylogenetic variation of two.b.pls. It might be of interest with 3+ modules to perform separate phylogenetic integration tests between all pairwise comparisons of modules. This can be done, test by test, and the levels of integration can be compared with compare.pls. Such results are different than using the average amount of integration when more than two modules are input, as found with phylo.integration.

Notes for geomorph 3.0.4 and subsequent versions:

Compared to previous versions of geomorph, users might notice differences in effect sizes. Previous versions used z-scores calculated with expected values of statistics from null hypotheses (sensu Collyer et al. 2015); however Adams and Collyer (2016) showed that expected values for some statistics can vary with sample size and variable number, and recommended finding the expected value, empirically, as the mean from the set of random outcomes. Geomorph 3.0.4 and subsequent versions now center z-scores on their empirically estimated expected values and where appropriate, log-transform values to assure statistics are normally distributed. This can result in negative effect sizes, when statistics are smaller than expected compared to the average random outcome. For ANOVA-based functions, the option to choose among different statistics to measure effect size is now a function argument.

Value

Objects of class "pls" from integration.test return a list of the following:

`r.pls`	The estimate of morphological integration: PLS.corr. The mean of pairwise PLS correlations between partitions is used when there are more than two partitions.
`r.pls.mat`	The pairwise r.pls, if the number of partitions is greater than 2.
`P.value`	The empirically calculated P-value from the resampling procedure.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`left.pls.vectors`	The singular vectors of the left (x) block (for 2 modules only).
`right.pls.vectors`	The singular vectors of the right (y) block (for 2 modules only).
`random.r`	The correlation coefficients found in each random permutation of the resampling procedure.
`XScores`	Values of left (x) block projected onto singular vectors (for 2 modules only).
`YScores`	Values of right (y) block projected onto singular vectors (for 2 modules only).
`svd`	The singular value decomposition of the cross-covariances (for 2 modules only).
`A1`	Input values for the left block (for 2 modules only).
`A2`	Input values for the right block (for 2 modules only).
`A1.matrix`	Left block (matrix) found from A1 (for 2 modules only).
`A2.matrix`	Right block (matrix) found from A2 (for 2 modules only).
`Pcov`	The phylogenetic transformation matrix, needed for certain other analyses.
`permutations`	The number of random permutations used in the resampling procedure.
`call`	The match call.

Author(s)

Dean Adams

References

Adams, D.C. and R. Felice. 2014. Assessing phylogenetic morphological integration and trait covariation in morphometric data using evolutionary covariance matrices. PLOS ONE. 9(4):e94335.

Collyer, M.L., D.J. Sekora, and D.C. Adams. 2015. A method for analysis of phenotypic change for phenotypes described by high-dimensional data. Heredity. 115:357-365.

Adams, D.C. and M.L. Collyer. 2016. On the comparison of the strength of morphological integration across morphometric datasets. Evolution. 70:2623-2631.

Adams, D.C. and M.L. Collyer. 2018. Multivariate comparative methods: evaluations, comparisons, and recommendations. Systematic Biology. 67:14-31.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

See Also

integration.test, modularity.test, and two.b.pls

Examples

```
data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment
land.gps<-c("A","A","A","A","A","B","B","B","B","B","B")

IT<- phylo.integration(Y.gpa$coords,partition.gp=land.gps,
  phy=plethspecies$phy,iter=999)
summary(IT) # Test summary
plot(IT) # PLS plot

 ### Visualize shape variation using picknplot.shape Because picknplot
 ### requires user decisions, the following example
 ### is not run (but can be with removal of #).
 ### For detailed options, see the picknplot help file
 # picknplot.shape(plot(IT))
```

---

phylo.modularity *Evaluate the degree of phylogenetic modular signal in Procrustes shape variables*

Description

Function quantifies the degree of modularity between two or more hypothesized modules of Procrustes shape variables in a phylogenetic context and compares this to patterns found by randomly assigning landmarks into subsets

Usage

```
phylo.modularity(
  A,
  partition.gp,
  phy,
  CI = FALSE,
  iter = 999,
  seed = NULL,
  print.progress = TRUE
)
```

Arguments

A	A 3D array (p x k x n) containing Procrustes shape variables for all specimens, or a matrix (n x variables)
partition.gp	A list of which landmarks (or variables) belong in which partition: (e.g. A, A, A, B, B, B, C, C, C)
phy	A phylogenetic tree of class phylo - see read.tree in library ape
CI	A logical argument indicating whether bootstrapping should be used for estimating confidence intervals
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function quantifies the degree of phylogenetic modularity in two or more hypothesized modules of Procrustes shape variables as defined by landmark coordinates, under a Brownian motion model of evolution. The degree of modularity is characterized by the covariance ratio covariance ratio (CR: see Adams 2016). The phylogenetic version of the approach procedure utilizes the evolutionary covariance matrix among traits found under a Brownian motion model of evolution as the basis of the analysis. This is the same matrix used to evaluate patterns of phylogenetic morphological integration as described in Adams and Felice (2014).

Input may be either a 2D matrix of phenotypic values, or a 3D array of Procrustes shape variables. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. The degree of modularity is quantified using the CR coefficient (Adams 2016). If more than two modules are defined, the average pairwise CR coefficient is utilized. The CR coefficient for the observed modular hypothesis is then compared to a distribution of values obtained by randomly assigning landmarks into subsets, with the restriction that the number of landmarks in each subset is identical to that observed in each of the original partitions. A significant

modular signal is found when the observed CR coefficient is small relative to this distribution (see Adams 2016). Such a result implies that there is significantly greater independence among modules than is expected under the null hypothesis of random associations of variables (neither modular nor integrated structure). This result is consistent with the identification of significant modular structure in the data. For landmark data, the CR coefficient found from the average CR across a 90 degree rotation of the data is used as the test statistic (see Adams 2016). In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019).

Value

Objects of class "CR" from modularity.test return a list of the following:

`CR`	Covariance ratio: The estimate of the observed modular signal.
`CInterval`	The bootstrapped 95 percent confidence intervals of the CR, if CI = TRUE.
`CR.boot`	The bootstrapped CR values, if CI = TRUE. For more than two partitions, this is the mean CR of pairwise CRs.
`P.value`	The empirically calculated P-value from the resampling procedure.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`CR.mat`	For more than two partitions, the pairwise CRs among partitions.
`random.CR`	The CR calculated in each of the random permutations of the resampling procedure.
`Pcov`	The phylogenetic transformation matrix, needed for certain other analyses.
`permutations`	The number of random permutations used in the resampling procedure.
`call`	The match call.

Author(s)

Dean Adams

References

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.

Adams, D.C. and R. Felice. 2014. Assessing phylogenetic morphological integration and trait covariation in morphometric data using evolutionary covariance matrices. PLOS ONE. 9(4):e94335.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

Examples

```
data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment
land.gps<-c("A","A","A","A","A","B","B","B","B","B","B")

MT <- phylo.modularity(Y.gpa$coords, partition.gp=land.gps,
```

```
phy=plethspecies$phy,
CI = FALSE, iter=499)
summary(MT) # Test summary
plot(MT) # Histogram of CR sampling distribution
```

---

physignal *Assessing phylogenetic signal in Procrustes shape variables*

Description

Function calculates the degree of phylogenetic signal from Procrustes shape variables

Usage

```
physignal(A, phy, iter = 999, seed = NULL, print.progress = FALSE)
```

Arguments

A	A matrix (n x [p x k]) or 3D array (p x k x n) containing Procrustes shape variables for a set of specimens
phy	A phylogenetic tree of class phylo - see read.tree in library ape
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function estimates the degree of phylogenetic signal present in Procrustes shape variables for a given phylogeny. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. The degree of phylogenetic signal in data is estimated using the multivariate version of the K-statistic (Kmult: Adams 2014). This value evaluates the degree of phylogenetic signal in a dataset relative to what is expected under a Brownian motion model of evolution. For geometric morphometric data, the approach is a mathematical generalization of the Kappa statistic (Blomberg et al. 2003) appropriate for highly multivariate data (see Adams 2014).Significance testing is found by permuting the shape data among the tips of the phylogeny. In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2019). Values from these distributions are log-transformed prior to effect size estimation, to assure normally distributed data.

This function can also be used with univariate data (i.e. centroid size) if imported as matrix with rownames giving the taxa names. In this case, the estimate of phylogenetic signal is identical to that found using the standard kappa statistic (Blomberg et al. 2003).

The generic functions, `print`, `summary`, and `plot` all work with `physignal`. The generic function, `plot`, produces a histogram of random K statistics, associated with the resampling procedure.

Notes for geomorph 3.0:

Compared to older versions of geomorph, the order of input variables has changed, so that it is consistent with other functions in the program. Additionally, users should note that the function physignal no longer contains multiple methods. Only Kmult is used. Thus, for older scripts method="" should be removed from the function call.

Value

Function returns a list with the following components:

`phy.signal`	The estimate of phylogenetic signal.
`pvalue`	The significance level of the observed signal.
`Effect.Size`	The multivariate effect size associated with sigma.d.ratio.
`random.K`	Each random K-statistic from random permutations.
`permutations`	The number of random permutations used in the resampling procedure.
`PACA`	A phylogenetically aligned component analysis, based on OLS residuals.
`K.by.p`	The phylogenetic signal in 1, 1:2, 1:3, ..., 1:p dimensions, for the p components from PACA.
`call`	The matched call

Author(s)

Dean Adams and Michael Collyer

References

Blomberg SP, Garland T, Ives AR. 2003. Testing for phylogenetic signal in comparative data: behavioral traits are more labile. Evolution, 57:717-745.

Adams, D.C. 2014. A generalized K statistic for estimating phylogenetic signal from shape and other high-dimensional multivariate data. Systematic Biology. 63:685-697.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

See Also

`gm.prcomp`

Examples

```
data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment

#Test for phylogenetic signal in shape
PS.shape <- physignal(A=Y.gpa$coords,phy=plethspecies$phy,iter=999)
summary(PS.shape)
plot(PS.shape)
plot(PS.shape$PACA, phylo = TRUE)
PS.shape$K.by.p # Phylogenetic signal profile

#Test for phylogenetic signal in size
PS.size <- physignal(A=Y.gpa$Csize,phy=plethspecies$phy,iter=999)
summary(PS.size)
plot(PS.size)
```

---

picknplot.shape *Pick points in geomorph scatterplots to visualize shape variation*

Description

Function plots the shape corresponding to a clicked point in the area of a geomorph plot

Usage

```
picknplot.shape(x, ...)
```

Arguments

x	a geomorph plot object of class plot.gm.prcomp, plot.procD.lm, plot.pls, or plotAllometry
...	other arguments passed to plotRefToTarget

Details

THIS FUNCTION IS A BIT EXPERIMENTAL!

This function recycles plots generated by plot.gm.prcomp, plot.procD.lm, plot.pls, or plotAllometry, and makes them interactive to visualize shape variation by selecting one or more points in morphospace. The function uses shape.predictor to estimate the shape corresponding to the selected point(s) based on the prediction underlying the scatterplot, and it plots the estimated shape as compared to the consensus landmark configuration using plotRefToTarget. The user is then prompted as to whether the plotted shape is to be saved as a png file, in which case the name of the file needs to be provided (without quotation marks). Interactive plots are at present available for plots produced by plot.gm.prcomp. The function is limited in terms of the options for plotRefToTarget (because of the complexity of graphics); using shape.predictor and plotRefToTarget, directly, will always offer more flexibility.

IF YOU EXPERIENCE AN ERROR, please use shape.predictor and plotRefToTarget, directly. (But please alert the geomorph package maintainer.)

Value

A list with the following components:

points	A list with the xy coordinates of the selected points.
shapes	A list with the corresponding estimated shapes.

Author(s)

Antigoni Kaliontzopoulou, Emma Sherratt, & Michael Collyer

See Also

shape.predictor, plotRefToTarget

rgl-package (used in 3D plotting)

Examples

```
### Because picknplot requires user decisions, the following examples
### are not run (but can be with removal of #s)

# 2d
# data(plethodon)
# Y.gpa <- gpagen(plethodon$land)
# pleth.pca <- gm.prcomp(Y.gpa$coords)
# pleth.pca.plot <- plot(pleth.pca)
# picknplot.shape(pleth.pca.plot)
# May change arguments for plotRefToTarget
# picknplot.shape(plot(pleth.pca), method = "points", mag = 3,
# links=plethodon$links)

# 2d with phylogeny
# data(plethspecies)
# Y.gpa <- gpagen(plethspecies$land)
# gps <- as.factor(c(rep("gp1", 5), rep("gp2", 4))) # Two random groups
# pleth.phylo <- gm.prcomp(Y.gpa$coords, plethspecies$phy)
# pleth.phylomorphospace <- plot(pleth.phylo, phylo = TRUE, cex = 2,
# pch = 22, bg = gps, phylo.par = list(edge.color = "blue",
# edge.width = 2, edge.lty = 2,
# node.pch = 22, node.bg = "black"))
# links.species <- plethodon$links[-11,]
# links.species[11, 1] <- 11
# picknplot.shape(pleth.phylomorphospace, method = "points",
# links = links.species)

# 2d allometry
# gdf <- geomorph.data.frame(Y.gpa, site = plethodon$site,
# species = plethodon$species)
# fit <- procD.lm(coords ~ log(Csize), data=gdf, iter=0,
# print.progress = FALSE)
# Predline
```

```
# PA <- plotAllometry(fit, size = gdf$Csize, logsz = TRUE,
# method = "PredLine", pch = 19)
# picknplot.shape(PA)

# 3d and two-b-pls
# data("scallops")
# Y.gpa <- gpagen(scallops$coorddata, curves = scallops$curvslide,
#              surfaces = scallops$surfslide)
# PLS <- two.b.pls(Y.gpa$coords, Y.gpa$Csize)
# PLS.plot = plot(PLS)
# picknplot.shape(PLS.plot)
```

`plethodon` *Landmark data from Plethodon salamander heads*

Description

Landmark data from Plethodon salamander heads

Author(s)

Dean Adams

References

Adams, D. C. 2004. Character displacement via aggressive interference in Appalachian salamanders. Ecology. 85:2664-2670.

Adams, D.C. 2010. Parallel evolution of character displacement driven by competitive selection in terrestrial salamanders. BMC Evolutionary Biology. 10(72)1-10.

`plethShapeFood` *Head shape and food use data from Plethodon salamanders*

Description

Head shape and food use data from Plethodon salamanders

Author(s)

Dean Adams

References

Adams, D. C., and F. J. Rohlf. 2000. Ecological character displacement in Plethodon: biomechanical differences found from a geometric morphometric study. Proceedings of the National Academy of Sciences, U.S.A. 97:4106-4111

`plethspecies` *Head shape and phylogenetic relationships for several Plethodon salamander species*

Description

Head shape and phylogenetic relationships for several Plethodon salamander species

Author(s)

Dean Adams

References

Phylogeny pruned from: Wiens et al. (2006). Evol.

Data from: Adams and Rohlf (2000); Adams et al. (2007); Arif et al. (2007) Myers and Adams (2008)

`plot.bilat.symmetry` *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'bilat.symmetry'
plot(x, warpgrids = TRUE, mesh = NULL, ...)
```

Arguments

`x`	plot object (from `bilat.symmetry`)
`warpgrids`	Logical argument whether to include warpgrids
`mesh`	Option to include mesh in warpgrids plots
`...`	other arguments passed to plot

Author(s)

Michael Collyer

plot.CR *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'CR'
plot(x, ...)
```

Arguments

x	plot object (from phylo.modularity)
...	other arguments passed to plot

Author(s)

Michael Collyer

plot.CR.phylo *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'CR.phylo'
plot(x, ...)
```

Arguments

x	plot object (from phylo.modularity)
...	other arguments passed to plot

Author(s)

Dean Adams

plot.evolrate *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'evolrate'
plot(x, ...)
```

Arguments

x	plot object
...	other arguments passed to plot

Author(s)

Michael Collyer

plot.gm.prcomp *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'gm.prcomp'
plot(
  x,
  axis1 = 1,
  axis2 = 2,
  flip = NULL,
  phylo = FALSE,
  time.plot = FALSE,
 phylo.par = list(tip.labels = TRUE, node.labels = TRUE, anc.states = TRUE, node.pch =
  21, node.bg = "grey", node.cex = 1, edge.color = "black", edge.width = 1, tip.txt.cex
    = 1, tip.txt.col = "black", tip.txt.adj = c(-0.1, -0.1), node.txt.cex = 1,
    node.txt.col = "grey", node.txt.adj = c(-0.1, -0.1)),
  ...
)
```

Arguments

x	An object of class gm.prcomp
axis1	A value indicating which PC axis should be displayed as the X-axis (default = PC1)
axis2	A value indicating which PC axis should be displayed as the Y-axis (default = PC2)
flip	An argument that if not NULL can be used to flip components in the plot. The values need to match axis1 or axis2. For example, if axis1 = 3 and axis2 = 4, flip = 1 will not change either axis; flip = 3 will flip only the horizontal axis; flip = c(3, 4) will flip both axes.
phylo	A logical value indicating whether the phylogeny should be projected to PC space
time.plot	A logical value indicating if a 3D plot with the phylogeny and time as the z-axis is desired
phylo.par	A list of plotting parameters for the inclusion of a phylogeny, including: logicals for whether features should be included (tip.labels, nodel.labels, anc.states), toggled as TRUE/FALSE; edge parameters (edge.color, edge.width, edge.lty); node parameters (node.bg, node.pch, node.cex); and label parameters (tip.txt.cex, tip.txt.col, tip.txt.adj, node.txt.cex, node.txt.col, node.txt.adj).
...	other arguments passed to plot. For plots with a phylogeny, these parameters pertain to the tip values.

Value

An object of class "plot.gm.prcomp" is a list with components that can be used in other plot functions, such as the type of plot, points, a group factor, and other information depending on the plot parameters used. A time plot is an addendum to the normal 2D plot, and does not add additional output.

NOTE: To visualize shape variation across PC axes in 2d plots, use picknplot.shape.

Author(s)

Antigoni Kaliontzopoulou, Michael Collyer

See Also

plotRefToTarget picknplot.shape

plot.gpagen *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'gpagen'
plot(x, ...)
```

Arguments

x	plot object (from gpagen)
...	other arguments passed to plotAllSpecimens

Author(s)

Michael Collyer

plot.mshape *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'mshape'
plot(x, links = NULL, ...)
```

Arguments

x	plot object (from mshape)
links	An optional matrix defining for links between landmarks
...	other arguments passed to plot

Author(s)

Antigoni Kaliontzopoulou

See Also

define.links

plot.physignal *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'physignal'
plot(x, ...)
```

Arguments

x	plot object (from physignal)
...	other arguments passed to plot

Author(s)

Michael Collyer

plot.pls *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'pls'
plot(x, label = NULL, ...)
```

Arguments

x	plot object (from phylo.integration or two.b.pls)
label	Optional vector to label points
...	other arguments passed to plot. The function returns values that can be used with picknplot.shape (in a limited capacity). In most cases, greater flexibility can be attained with using plotRefToTarget and shape.predictor.

Value

If shapes = TRUE, function returns a list containing the shape coordinates of the extreme ends of axis1 and axis2 if 3D arrays were originally provided for each

Author(s)

Michael Collyer

`plot.procD.lm` *Plot Function for geomorph*

Description

Plot Function for geomorph

Usage

```
## S3 method for class 'procD.lm'
plot(
  x,
  type = c("diagnostics", "regression", "PC"),
  outliers = FALSE,
  predictor = NULL,
  reg.type = c("PredLine", "RegScore"),
  ...
)
```

Arguments

`x`	plot object (from `procD.lm`)
`type`	Indicates which type of plot, choosing among diagnostics, regression, or principal component plots. Diagnostic plots are similar to `lm` diagnostic plots, but for multivariate data. Regression plots plot multivariate dispersion in some fashion against predictor values. PC plots project data onto the eigenvectors of the covariance matrix for fitted values.
`outliers`	Logical argument to include outliers plot, if diagnostics are performed
`predictor`	An optional vector if "regression" plot type is chosen, and is a variable likely used in `procD.lm`. This vector is a vector of covariate values equal to the number of observations.
`reg.type`	If "regression" is chosen for plot type, this argument indicates whether a prediction line (Predline) plot, or regression score (RegScore) plotting is performed.
`...`	other arguments passed to plot (helpful to employ different colors or symbols for different groups). See `plot.default` and `par`

Value

An object of class "plot.procD.lm" is a list with components that can be used in other plot functions, such as the type of plot, points, a group factor, and other information depending on the plot parameters used.

Author(s)

Michael Collyer

`plotAllometry` *Plotting to assist visualization of shape-size covariation (allometry)*

Description

Function performs plotting for a procD.lm fit and a vector of size measures.

Usage

```
plotAllometry(
  fit,
  size,
  logsz = TRUE,
  method = c("PredLine", "RegScore", "size.shape", "CAC"),
  ...
)
```

Arguments

`fit`	A procD.lm fit.
`size`	A vector of the same length as the number of observations in the fit.
`logsz`	A logical value to indicate whether to first find the logarithm of size.
`method`	The method of allometric visualization, which includes CAC, PredLine, RegScore, and size.shape (PCA)
`...`	Other arguments passed on to plot.default

Details

Prior to geomorph 3.0.0, the function, plotAllometry, was used to perform linear regression of shape variables and size, and produce plots to visualize shape allometries. This function was deprecated when procD.allometry was launched with geomorph 3.0.0, which performed homogeneity of slopes tests to determine if a common allometry or unique group allometries were more appropriate as a model. The S3 generic, plot.procD.allometry provided the same plotting as plotAllometry before it. In geomorph 3.1.0, procD.allometry has been deprecated in favor of using `procD.lm` and `pairwise` for analyses, which can include additional variables, thus eliminating plot.procD.allometry. This function coalesces a few plotting options found in other functions, as a wrapper, for the purpose of retaining the plot.procD.allometry options in one place.

There are fundamentally two different kinds of allometry plots: those based on linear models and those that do not have a linear model basis (more detail below). The common allometric component (CAC) and size-shape PCA (Mitteroecker et al. 2004) are plotting strategies that do not have results that vary with linear model parameters. By contrast, prediction lines (PredLine, Adams and Nistri 2010) and regression scores (RegScore, Drake and Klingenberg 2008) are based on fitted values and

regression coefficients, respectively, to visualize allometric patterns. The plotAllometry function will extract necessary components from a procD.lm fit to calculate these various statistics (although the variables used in the procD.lm fit are inconsequential for CAC and size-shape PCA; only the shape variables are used).

There are multiple ways to visualize allometry. One way is to simply append a size variable to shape variables and perform a principal component analysis (PCA). In the event that size and shape strongly covary, the first PC scores might reflect this (Mitteroecker et al. 2004). Alternatively, the major axis of covariation between size and shape can be found by a singular value decomposition of their cross-products, a process known as two-block partial least squares (PLS; Rohlf and Corti 2000). This major axis of variation is often referred to as the common allometric component (CAC; Mitteroecker et al. 2004). Neither of these methods is associated with a model of allometric shape change, especially as such change might vary for different groups. As such, these methods have limited appeal for comparing group allometries (although color-coding groups in plots might reveal different trends in the plot scatter).

By contrast, describing a linear model (with procD.lm) that has an explicit definition of how shape allometries vary by group can be more informative. The following are the three most general models:

simple allometry: shape ~ size

common allometry, different means: shape ~ size + groups

unique allometries: shape ~ size * groups

However, other covariates can be added to these models. One could define these models with procD.lm and use anova.lm.rrpp to explicitly test which model is most appropriate. The function, pairwise can also be used to test pairwise differences among least-squares means or slopes. To visualize different allometric patterns, wither prediction lines (PredLine; Adams and Nistri 2010) or regression scores (RegScore; Drake and Klingenberg 2008) can be used. The former plots first PCs of fitted values against size; the latter calculates a regression score as a projection of data on normalized vector that expresses the covariation between shape and the regression coefficients for size, conditioned on other model effects. For a simple allometry model, CAC and RegScore are the same (Adams et al. 2013) but RegScore, like PredLine but unlike CAC, generalizes to complex models. Either PredLine or RegScore can help elucidate divergence in allometry vectors among groups.

If the variable for size is used in the procD.lm fit, the plot options will resemble past allometry plots found in geomorph. However, with this updated function philosophy, the model fit does not have to necessarily contain size. This might be useful if one wishes to visualize whether shape, size, and some other variable covary in some way (by first performing a procD.lm fit between shape and another covariate, then performing plotAllometry with that fit and size). For example, one can entertain the question, "Are species differences in shape merely a manifestation of shape allometry, when species differ in size?" By fitting a model, shape ~ species, then using plotAllometry for the model fit (with either PredLine or RegScore), the plot will help reveal if allometry and species effects are confounded.

The following are brief descriptions of the different plotting methods, with references.

- If "method = PredLine" (the default) the function calculates fitted values from a procD.lm fit, and plots the first principal component of the "predicted" values versus size as a stylized graphic of the allometric trend (Adams and Nistri 2010). This method is based on linear models and can allow for other model variable to be incorporated.

- If "method = RegScore" the function calculates standardized shape scores from the regression of shape on size, and plots these versus size (Drake and Klingenberg 2008). For a single allometry, these shape scores are mathematically identical to the CAC (Adams et al. 2013). This method is based on linear models and can allow for other model variable to be incorporated.
- If "method = size.shape" the function perform principal components analysis on a data space containing both shape and size (sensu Mitteroecker et al. 2004). This method is not based on linear models and results will not be changed by changing the allometry model.
- If "method = CAC" the function calculates the common allometric component of the shape data, which is an estimate of the average allometric trend for group-mean centered data (Mitteroecker et al. 2004). The function also calculates the residual shape component (RSC) for the data. This method is not based on linear models and results will not be changed by changing the allometry model.

The function returns values that can be used with `picknplot.shape` or `shape.predictor` and `plotRefToTarget` to visualize shape changes in the plot.

Value

An object of class plotAllometry returns some combination of CAC values, the residual shape component (RSC, associated with CAC approach), PredLine values, RegScore values, PC points for the size-shape PCA, and PCA statistics, depending on arguments used. The size variable and GM statistics from the original model fit are also returned. .

Author(s)

Michael Collyer

References

Adams, D. C., and A. Nistri. 2010. Ontogenetic convergence and evolution of foot morphology in European cave salamanders (Family: Plethodontidae). BMC Evol. Biol. 10:1-10.

Adams, D.C., F.J. Rohlf, and D.E. Slice. 2013. A field comes of age: geometric morphometrics in the 21st century. Hystrix. 24:7-14.

Drake, A. G., and C. P. Klingenberg. 2008. The pace of morphological change: Historical transformation of skull shape in St Bernard dogs. Proc. R. Soc. B. 275:71-76.

Mitteroecker, P., P. Gunz, M. Bernhard, K. Schaefer, and F. L. Bookstein. 2004. Comparison of cranial ontogenetic trajectories among great apes and humans. J. Hum. Evol. 46:679-698.

Rohlf, F.J., and M. Corti. 2000. The use of partial least-squares to study covariation in shape. Systematic Biology 49: 740-753.

Examples

```
# Simple allometry
data(plethodon)
Y.gpa <- gpagen(plethodon$land, print.progress = FALSE)    #GPA-alignment

gdf <- geomorph.data.frame(Y.gpa, site = plethodon$site,
```

```
species = plethodon$species)
fit <- procD.lm(coords ~ log(Csize), data=gdf, iter=0,
print.progress = FALSE)

# Predline
plotAllometry(fit, size = gdf$Csize, logsz = TRUE,
method = "PredLine", pch = 19)

# same as
logSize <- log(gdf$Csize)
plot(fit, type = "regression", reg.type = "PredLine",
predictor = logSize, pch = 19)

# RegScore
plotAllometry(fit, size = gdf$Csize, logsz = TRUE,
method = "RegScore", pch = 19)

# same as
plot(fit, type = "regression", reg.type = "RegScore",
predictor = logSize, pch = 19)

# CAC
plotAllometry(fit, size = gdf$Csize, logsz = TRUE,
method = "CAC", pch = 19)

# same (first plot) as
PLS <- two.b.pls(log(gdf$Csize), gdf$coords, print.progress = FALSE)
plot(PLS)

# Group Allometries
fit <- procD.lm(coords ~ Csize * species * site, data=gdf, iter=0,
print.progress = FALSE)

# CAC (should not change from last time; model change has no effect)
plotAllometry(fit, size = gdf$Csize, logsz = TRUE, method = "CAC",
pch = 19)

# Predline
plotAllometry(fit, size = gdf$Csize, logsz = TRUE, method = "PredLine",
pch = 19, col = as.numeric(interaction(gdf$species, gdf$site)))

# RegScore
plotAllometry(fit, size = gdf$Csize, logsz = TRUE, method = "RegScore",
pch = 19, col = as.numeric(interaction(gdf$species, gdf$site)))

# Size-Shape PCA

pc.plot <- plotAllometry(fit, size = gdf$Csize, logsz = TRUE,
method = "size.shape",
pch = 19, col = as.numeric(interaction(gdf$species, gdf$site)))
summary(pc.plot$size.shape.PCA)

# Are species' shape differences just a manifestation of shape allometry?
```

```
fit3 <- procD.lm(coords ~ species, data = gdf, iter = 0,
print.progress = FALSE)
plotAllometry(fit3, size = gdf$Csize, logsz = TRUE, method = "RegScore",
pch = 19, col = as.numeric(gdf$species))

# No evidence this is the case
```

`plotAllSpecimens` *Plot landmark coordinates for all specimens*

Description

Function plots landmark coordinates for a set of specimens

Usage

```
plotAllSpecimens(
  A,
  mean = TRUE,
  links = NULL,
  label = FALSE,
  plot.param = list()
)
```

Arguments

`A`	A 3D array (p x k x n) containing Procrustes shape variables for a set of specimens
`mean`	A logical value indicating whether the mean shape should be included in the plot
`links`	An optional matrix defining for links between landmarks (only if mean=TRUE)
`label`	A logical value indicating whether landmark numbers will be plotted (only if mean=TRUE)
`plot.param`	A list of plot parameters for the points (pt.bg, pt.cex), mean (mean.bg, mean.cex), links (link.col, link.lwd, link.lty) and landmark labels (txt.cex, txt.adj, txt.pos, txt.col)

Details

The function creates a plot of the landmark coordinates for all specimens. This is useful for examining patterns of variation in Procrustes shape variables, after a GPA has been performed. If "mean=TRUE", the mean shape will be calculated and added to the plot. Additionally, if a matrix of links is provided, the landmarks of the mean shape will be connected by lines. The link matrix is an m x 2 matrix, where m is the desired number of links. Each row of the link matrix designates the two landmarks to be connected by that link. The function will plot either two- or three-dimensional data (e.g. see `define.links`).

Author(s)

Dean Adams

See Also

rgl-package (used in 3D plotting)

Examples

```
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment

plotAllSpecimens(Y.gpa$coords,links=plethodon$links)
```

---

plotOutliers *Find potential outliers*

Description

Function plots all specimens ordered by distance from the mean.

Usage

```
plotOutliers(A, groups = NULL, inspect.outliers = FALSE)
```

Arguments

A	A 3D array (p x k x n) containing Procrustes shape variables for a set of specimens
groups	An optional factor defining groups
inspect.outliers	A logical value indicating whether to plot outlier shape configurations as compared to the consensus

Details

The function creates a plot of all specimens ordered by their Procrustes distance from the mean shape. The median distance (unbroken line) and upper and lower quartiles (dashed lines) summarize the distances from the mean shape. Specimens falling above the upper quartile are plotted in red. The user may optionally also inspect the shapes of identified outlier configurations as compared to the consensus, in order to identify digitization errors or other data issues. The addresses of all specimens are returned in the order displayed in the plot for further inspection by plotRefToTarget.

If the data have strong group structure and there is reasonable belief that the whole sample mean should not be used, then a factor defining the groups can be used.

Value

Function returns the landmark addresses of all specimens ordered as in the plot. If groups are used, function returns a list structure and a plot for each level in groups.

Author(s)

Emma Sherratt & Antigoni Kaliontzopoulou

See Also

gpagen

plotAllSpecimens

Examples

```
data(plethodon)
# let's make some outliers
newland <- plethodon$land
newland[c(1,8),,2] <- newland[c(8,1),,2]
newland[c(3,11),,26] <- newland[c(11,3),,2]
Y<- gpagen(newland) # GPA
out <- plotOutliers(Y$coords) # function returns dimnames and address
# of all specimens ordered
plotOutliers(Y$coords, inspect.outliers = TRUE) # function also produces
# plots of identified outlier specimens compared to the mean shape

# example with groups
plotOutliers(Y$coords, groups = plethodon$species,
inspect.outliers = TRUE)
```

---

plotRefToTarget *Plot shape differences between a reference and target specimen*

Description

Function plots shape differences between a reference and target specimen

Usage

```
plotRefToTarget(
  M1,
  M2,
  mesh = NULL,
  outline = NULL,
  method = c("TPS", "vector", "points", "surface"),
  mag = 1,
  links = NULL,
```

```
  label = FALSE,
  axes = FALSE,
  gridPars = NULL,
  useRefPts = FALSE,
  ...
)
```

Arguments

M1	Matrix of landmark coordinates for the first (reference) specimen
M2	Matrix of landmark coordinates for the second (target) specimen
mesh	A mesh3d object for use with method="surface"
outline	An x,y curve or curves warped to the reference (2D only)
method	Method used to visualize shape difference; see below for details
mag	The desired magnification to be used when visualizing the shape difference (e.g., mag=2)
links	An optional matrix defining for links between landmarks
label	A logical value indicating whether landmark numbers will be plotted
axes	A logical value indicating whether the box and axes should be plotted (points and vector only)
gridPars	An optional object made by gridPar
useRefPts	An option (logical value) to use reference configuration points rather than target configuration points (when method = "TPS")
...	Additional parameters not covered by gridPar to be passed to plot, plot3d or shade3d

Details

The function generates a plot of the shape differences of a target specimen relative to a reference specimen. The option mag allows the user to indicates the degree of magnification to be used when displaying the shape difference. The function will plot either two- or three-dimensional data.

For two-dimensional data and thin-plate spline deformation plots, the user may also supply boundary curves of the object, which will be deformed from the reference to the target specimen using the thin-plate spline. Such curves are often useful in describing the biological shape differences expressed in the landmark coordinates. Note that to utilize this option, a boundary curve from a representative specimen must first be warped to the reference specimen using warpRefOutline.

Additionally, if a matrix of links is provided, the landmarks will be connected by lines. The link matrix is an M x 2 matrix, where M is the desired number of links. Each row of the link matrix designates the two landmarks to be connected by that link.

Four distinct methods for plots are available:

1. TPS a thin-plate spline deformation grid is generated. For 3D data, this method will generate thin-plate spline deformations in the x-y and x-z planes.
2. vector: a plot showing the vector displacements between corresponding landmarks in the reference and target specimen is shown.

3. points a plot is displayed with the landmarks in the target overlaying those of the reference.
4. surface a mesh3d surface is warped using thin-plate spline (for 3D data only). Requires mesh3d object in option mesh, made using warpRefMesh.

This function combines numerous plotting functions found in Claude (2008).

Value

If using method="surface", function will return the warped mesh3d object.

Author(s)

Dean Adams, Emma Sherratt, Antigoni Kaliontzopoulou & Michael Collyer

References

Claude, J. 2008. Morphometrics with R. Springer, New York.

See Also

gridPar

define.links

warpRefMesh

warpRefOutline

rgl-package (used in 3D plotting)

Examples

```
# Two dimensional data
data(plethodon)
Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
ref<-mshape(Y.gpa$coords)
plotRefToTarget(ref,Y.gpa$coords[,,39])
plotRefToTarget(ref,Y.gpa$coords[,,39], mag=2, outline=plethodon$outline)
#magnify by 2X
plotRefToTarget(ref,Y.gpa$coords[,,39], method="vector", mag=3)
plotRefToTarget(ref,Y.gpa$coords[,,39], method="points",
outline=plethodon$outline)
plotRefToTarget(ref,Y.gpa$coords[,,39], method="vector",
outline=plethodon$outline, mag=2.5)
plotRefToTarget(ref,Y.gpa$coords[,,39],
gridPars=gridPar(pt.bg = "green", pt.size = 1),
method="vector",mag=3)

# Three dimensional data
# data(scallops)
# Y.gpa<-gpagen(A=scallops$coorddata, curves=scallops$curvslide,
# surfaces=scallops$surfslide)
# ref<-mshape(Y.gpa$coords)
# plotRefToTarget(ref,Y.gpa$coords[,,1],method="points")
```

```
# scallinks <- matrix(c(1,rep(2:16, each=2),1), nrow=16, byrow=TRUE)
# plotRefToTarget(ref,Y.gpa$coords[,,1],
# gridPars=gridPar(tar.pt.bg = "blue", tar.link.col="blue",
# tar.link.lwd=2), method="points", links = scallinks)
```

`plotspec` *Plot 3D specimen, fixed landmarks and surface semilandmarks*

Description

A function to plot three-dimensional (3D) specimen along with its landmarks.

Usage

```
plotspec(
  spec,
  digitspec,
  fixed = NULL,
  fixed.pt.col = "red",
  fixed.pt.size = 10,
  mesh.ptsize = 1,
  centered = FALSE,
  ...
)
```

Arguments

`spec`	An object of class shape3d/mesh3d, or matrix of 3D vertex coordinates.
`digitspec`	Name of data matrix containing 3D fixed and/or surface sliding coordinates.
`fixed`	Numeric The number of fixed template landmarks (listed first in digitspec)
`fixed.pt.col`	The color for plotting fixed template landmarks (if any)
`fixed.pt.size`	The size for plotting fixed template landmarks (if any)
`mesh.ptsize`	Numeric Size to plot the mesh points (vertices), e.g. 0.1 for dense meshes, 3 for sparse meshes
`centered`	Logical Whether the data matrix is in the surface mesh coordinate system (centered = FALSE) or if the data were collected after the mesh was centered (centered = TRUE)- see details.
`...`	additional parameters which will be passed to rgl-plot3d or rgl-points3d.

Details

Function to plot 3D specimens along with their digitized "fixed" landmarks and semilandmarks "surface sliders" and "curve sliders". If specimen is a 3D surface (class shape3d/mesh3d) mesh is plotted. For visualization purposes, 3D coordinate data collected using digit.fixed or digitsurface and buildtemplate prior to build 1.1-6 were centered by default. Therefore use this function with centered=TRUE. Data collected outside geomorph should be read using centered=FALSE. The function assumes the fixed landmarks are listed at the beginning of the coordinate matrix (digitspec).

This function is a wrapper for several functions in the rgl-package package. Although there is some allowance for arguments to be passed to rgl-package functions, some override of rgl-plot3d arguments is required. Errors that result from trying to pass rgl-plot3d or rgl-points3d arguments should inspire the user to find solutions with rgl-package core functions.

Author(s)

Erik Otarola-Castillo, Emma Sherratt, Antigoni Kaliontzopoulou, & Michael Collyer

See Also

warpRefMesh

read.ply

rgl-package (used in 3D plotting)

Examples

```
data(scallopPLY)
ply <- scallopPLY$ply
digitdat <- scallopPLY$coords
plotspec(spec = ply, digitspec = digitdat, fixed = 16,
centered = TRUE, fixed.pt.col = "red",
fixed.pt.size = 15, col = "blue", size = 5)
```

---

print.bilat.symmetry *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'bilat.symmetry'
print(x, ...)
```

Arguments

x	print/summary object (from bilat.symmetry)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

---

print.combined.set *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'combined.set'
print(x, ...)
```

Arguments

x	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

---

print.compare.CR *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'compare.CR'
print(x, ...)
```

Arguments

x	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

`print.compare.pls` *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'compare.pls'
print(x, ...)
```

Arguments

`x`	print/summary object
`...`	other arguments passed to print/summary

Author(s)

Michael Collyer

`print.CR` *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'CR'
print(x, ...)
```

Arguments

`x`	print/summary object (from `phylo.modularity`)
`...`	other arguments passed to print/summary

Author(s)

Michael Collyer

print.CR.phylo *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'CR.phylo'
print(x, ...)
```

Arguments

x	print/summary object (from phylo.modularity)
...	other arguments passed to print/summary

Author(s)

Dean Adams

print.evolrate *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'evolrate'
print(x, ...)
```

Arguments

x	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.evolrate1 *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'evolrate1'
print(x, ...)
```

Arguments

x	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.geomorphShapes *Print/Summary function for geomorph*

Description

Print/Summary function for geomorph

Usage

```
## S3 method for class 'geomorphShapes'
print(x, ...)
```

Arguments

x	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.gm.prcomp *Print/Summary function for geomorph*

Description

Print/Summary function for geomorph

Usage

```
## S3 method for class 'gm.prcomp'
print(x, ...)
```

Arguments

x	print/summary object
...	other arguments passed to print/summary

Author(s)

Antigoni Kaliontzopoulou

print.gpagen *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'gpagen'
print(x, ...)
```

Arguments

x	print/summary object (from gpagen)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.morphol.disparity *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'morphol.disparity'
print(x, ...)
```

Arguments

x	print/summary object (from morphol.disparity)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.physignal *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'physignal'
print(x, ...)
```

Arguments

x	print/summary object (from physignal)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.pls *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'pls'
print(x, ...)
```

Arguments

x	print/summary object (from phylo.integration or two.b.pls)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

print.procD.lm *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'procD.lm'
print(x, ...)
```

Arguments

x	print/summary object (from procD.lm)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

procD.lm *Procrustes ANOVA/regression for Procrustes shape variables*

Description

Function performs Procrustes ANOVA with permutation procedures to assess statistical hypotheses describing patterns of shape variation and covariation for a set of Procrustes shape variables

Usage

```
procD.lm(
  f1,
  iter = 999,
  seed = NULL,
  RRPP = TRUE,
  SS.type = c("I", "II", "III"),
  effect.type = c("F", "cohenf", "SS", "MS", "Rsq"),
  int.first = FALSE,
  Cov = NULL,
  turbo = TRUE,
  Parallel = FALSE,
  data = NULL,
  print.progress = FALSE,
  ...
)
```

Arguments

f1	A formula for the linear model (e.g., y~x1+x2)
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the re-sampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
RRPP	A logical value indicating whether residual randomization should be used for significance testing
SS.type	SS.type A choice between type I (sequential), type II (hierarchical), or type III (marginal) sums of squares and cross-products computations.
effect.type	One of "F", "SS", or "cohen", to choose from which random distribution to estimate effect size. (The option, "cohen", is for Cohen's f-squared values. The default is "F". Values are log-transformed before z-score calculation to assure normally distributed data.)
int.first	A logical value to indicate if interactions of first main effects should precede subsequent main effects

Cov	An optional covariance matrix that can be used for generalized least squares estimates of coefficients and sums of squares and cross-products (see Adams and Collyer 2018).
turbo	A logical value that if TRUE, suppresses coefficient estimation in every random permutation. This will affect subsequent analyses that require random coefficients (see coef.lm.rrpp) but might be useful for large data sets for which only ANOVA is needed.
Parallel	Either a logical value to indicate whether parallel processing should be used or a numeric value to indicate the number of cores to use in parallel processing via the parallel library. If TRUE, this argument invokes forking of all processor cores, except one. If FALSE, only one core is used. A numeric value directs the number of cores to use, but one core will always be spared.
data	A data frame for the function environment, see geomorph.data.frame
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.
...	Arguments not listed above that can passed on to lm.rrpp, like weights, offset.

Details

The function quantifies the relative amount of shape variation attributable to one or more factors in a linear model and estimates the probability of this variation ("significance") for a null model, via distributions generated from resampling permutations. Data input is specified by a formula (e.g., y~X), where 'y' specifies the response variables (Procrustes shape variables), and 'X' contains one or more independent variables (discrete or continuous). The response matrix 'y' can be either in the form of a two-dimensional data matrix of dimension (n x [p x k]), or a 3D array (p x n x k). It is assumed that -if the data based on landmark coordinates - the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. The names specified for the independent (x) variables in the formula represent one or more vectors containing continuous data or factors. It is assumed that the order of the specimens in the shape matrix matches the order of values in the independent variables. Linear model fits (using the lm function) can also be input in place of a formula. Arguments for lm can also be passed on via this function.

The function two.d.array can be used to obtain a two-dimensional data matrix from a 3D array of landmark coordinates; however this step is no longer necessary, as procD.lm can receive 3D arrays as dependent variables. It is also recommended that geomorph.data.frame is used to create and input a data frame. This will reduce problems caused by conflicts between the global and function environments. In the absence of a specified data frame, procD.lm will attempt to coerce input data into a data frame, but success is not guaranteed.

The function performs statistical assessment of the terms in the model using Procrustes distances among specimens, rather than explained covariance matrices among variables. With this approach, the sum-of-squared Procrustes distances are used as a measure of SS (see Goodall 1991). The observed SS are evaluated through permutation. In morphometrics this approach is known as a Procrustes ANOVA (Goodall 1991), which is equivalent to distance-based anova designs (Anderson 2001). Two possible resampling procedures are provided. First, if RRPP=FALSE, the rows of the matrix of shape variables are randomized relative to the design matrix. This is analogous to a 'full' randomization. Second, if RRPP=TRUE, a residual randomization permutation procedure is utilized (Collyer et al. 2015). Here, residual shape values from a reduced model are obtained, and are randomized with respect to the linear model under consideration. These are then added to predicted

values from the remaining effects to obtain pseudo-values from which SS are calculated. NOTE: for single-factor designs, the two approaches are identical. However, when evaluating factorial models it has been shown that RRPP attains higher statistical power and thus has greater ability to identify patterns in data should they be present (see Anderson and terBraak 2003).

Effect-sizes (Z scores) are computed as standard deviates of either the SS, F, or Cohen's f-squared sampling distributions generated, which might be more intuitive for P-values than F-values (see Collyer et al. 2015). Values from these distributions are log-transformed prior to effect size estimation, to assure normally distributed data. The SS type will influence how Cohen's f-squared values are calculated. Cohen's f-squared values are based on partial eta-squared values that can be calculated sequentially or marginally, as with SS.

In the case that multiple factor or factor-covariate interactions are used in the model formula, one can specify whether all main effects should be added to the model first, or interactions should precede subsequent main effects (i.e., Y ~ a + b + c + a:b + ..., or Y ~ a + b + a:b + c + ..., respectively.)

The generic functions, `print`, `summary`, and `plot` all work with `procD.lm`. The generic function, `plot` has several options for plotting, using `plot.procD.lm`. Diagnostics plots, principal component plots (rotated to first PC of covariance matrix of fitted values), and regression plots can be performed. One must provide a linear predictor, and can choose among predicted values (PredLine) or regression scores (RegScore). In these plotting options, the predictor does not need to be size, and fitted values and residuals from the procD.lm fit are used rather than mean-centered values.

Notes for geomorph 3.1.0 and subsequent versions:

The procD.lm function is now a wrapper for the `lm.rrpp` function in the `RRPP` package. Examples below illustrate how to utilize `RRPP` functions along with `geomorph` functions for procD.lm objects, increasing the breadth of possible downstream analyses.

An important update in version 3.1.0 is that advanced.procD.lm and nested.update have been deprecated. The examples emphasize how pairwise comparisons can now be accomplished with `pairwise` and ANOVA updates for nested factors can be made with the `anova.lm.rrpp`, utilizing the error argument. These functions work on procD.lm objects that have already been created.

Notes for geomorph 3.0.6 and subsequent versions:

Compared to previous versions, GLS computations in random permutations are now possible in procD.lm. One should use RRPP = TRUE if a covariance matrix is provided as an argument. The method of SS calculations follows Adams and Collyer 2018. Additional output with a "gls." prefix is also available.

Notes for geomorph 3.0.4 and subsequent versions:

Compared to previous versions of geomorph, users might notice differences in effect sizes. Previous versions used z-scores calculated with expected values of statistics from null hypotheses (sensu Collyer et al. 2015); however Adams and Collyer (2016) showed that expected values for some statistics can vary with sample size and variable number, and recommended finding the expected value, empirically, as the mean from the set of random outcomes. Geomorph 3.0.4 and subsequent versions now center z-scores on their empirically estimated expected values and where appropriate, log-transform values to assure statistics are normally distributed. This can result in negative effect sizes, when statistics are smaller than expected compared to the average random outcome. For ANOVA-based functions, the option to choose among different statistics to measure effect size is now a function argument.

Value

An object of class "procD.lm" is a list containing the following

aov.table	An analysis of variance table; the same as the summary.
call	The matched call.
coefficients	A vector or matrix of linear model coefficients.
Y	The response data, in matrix form.
X	The model matrix.
QR	The QR decompositions of the model matrix.
fitted	The fitted values.
residuals	The residuals (observed responses - fitted responses).
weights	The weights used in weighted least-squares fitting. If no weights are used, NULL is returned.
Terms	The results of the terms function applied to the model matrix
term.labels	The terms used in constructing the aov.table.
data	The data frame for the model.
SS	The sums of squares for each term, model residuals, and the total.
SS.type	The type of sums of squares. One of type I or type III.
df	The degrees of freedom for each SS.
R2	The coefficient of determination for each model term.
F	The F values for each model term.
permutations	The number of random permutations (including observed) used.
random.SS	A matrix of random SS found via the resampling procedure used.
random.F	A matrix or vector of random F values found via the resampling procedure used.
random.cohenf	A matrix or vector of random Cohen's f-squared values found via the resampling procedure used.
permutations	The number of random permutations (including observed) used.
effect.type	The distribution used to estimate effect-size.
perm.method	A value indicating whether "Raw" values were shuffled or "RRPP" performed.
gls	This prefix will be used if a covariance matrix is provided to indicate GLS computations.

Author(s)

Dean Adams and Michael Collyer

References

Anderson MJ. 2001. A new method for non-parametric multivariate analysis of variance. Austral Ecology 26: 32-46.

Anderson MJ. and C.J.F. terBraak. 2003. Permutation tests for multi-factorial analysis of variance. Journal of Statistical Computation and Simulation 73: 85-113.

Goodall, C.R. 1991. Procrustes methods in the statistical analysis of shape. Journal of the Royal Statistical Society B 53:285-339.

Collyer, M.L., D.J. Sekora, and D.C. Adams. 2015. A method for analysis of phenotypic change for phenotypes described by high-dimensional data. Heredity. 115:357-365.

Adams, D.C. and M.L. Collyer. 2016. On the comparison of the strength of morphological integration across morphometric datasets. Evolution. 70:2623-2631.

Adams, D.C. and M.L. Collyer. 2018. Multivariate comparative methods: evaluations, comparisons, and recommendations. Systematic Biology. 67:14-31.

See Also

procD.pgls and lm.rrpp for more on linear model fits with RRPP. See RRPP-package for further details on functions that can use procD.lm objects.

Examples

```
## Not run:
### ANOVA example for Goodall's F test (multivariate shape vs. factors)
data(plethodon)
Y.gpa <- gpagen(plethodon$land)    #GPA-alignment
gdf <- geomorph.data.frame(Y.gpa,
site = plethodon$site,
species = plethodon$species) # geomorph data frame

fit1 <- procD.lm(coords ~ species * site,
data = gdf, iter = 999, turbo = TRUE,
RRPP = FALSE, print.progress = FALSE) # randomize raw values
fit2 <- procD.lm(coords ~ species * site,
data = gdf, iter = 999, turbo = TRUE,
RRPP = TRUE, print.progress = FALSE) # randomize residuals

summary(fit1)
summary(fit2)

# RRPP example applications

coef(fit2)
coef(fit2, test = TRUE)
anova(fit2) # same as summary
anova(fit2, effect.type = "Rsq")
# if species and site were modeled as random effects ...
anova(fit2, error = c("species:site", "species:site", "Residuals"))
# not run, because it is a large object to print
# remove # to run
```

```
# predict(fit2)

# TPS plots for fitted values of some specimens

M <- Y.gpa$consensus
plotRefToTarget(M, fit2$GM$fitted[,,1], mag = 3)
plotRefToTarget(M, fit2$GM$fitted[,,20], mag = 3)

### THE FOLLOWING ILLUSTRATES SIMPLER SOLUTIONS FOR
### THE NOW DEPRECATED advanced.procD.lm FUNCTION AND
### PERFORM ANALYSES ALSO FOUND VIA THE morphol.disparity FUNCTION
### USING THE pairwise FUNCTION

# Comparison of LS means, with log(Csize) as a covariate

# Model fits
fit.null <- procD.lm(coords ~ log(Csize) + species + site, data = gdf,
iter = 999, print.progress = FALSE, turbo = TRUE)
fit.full <- procD.lm(coords ~ log(Csize) + species * site, data = gdf,
iter = 999, print.progress = FALSE, turbo = TRUE)

# ANOVA, using anova.lm.rrpp function from the RRPP package
# (replaces advanced.procD.lm)
anova(fit.null, fit.full, print.progress = FALSE)

# Pairwise tests, using pairwise function from the RRPP package
gp <-  interaction(gdf$species, gdf$site)

PW <- pairwise(fit.full, groups = gp, covariate = NULL)

# Pairwise distances between means, summarized two ways
# (replaces advanced.procD.lm):
summary(PW, test.type = "dist", confidence = 0.95, stat.table = TRUE)
summary(PW, test.type = "dist", confidence = 0.95, stat.table = FALSE)

# Pairwise comaprisons of group variances, two ways
# (same as morphol.disaprity):
summary(PW, test.type = "var", confidence = 0.95, stat.table = TRUE)
summary(PW, test.type = "var", confidence = 0.95, stat.table = FALSE)
morphol.disparity(fit.full, groups = gp, iter = 999)

### Regression example
data(ratland)
rat.gpa<-gpagen(ratland)         #GPA-alignment
gdf <- geomorph.data.frame(rat.gpa) # geomorph data frame is easy
# without additional input

fit <- procD.lm(coords ~ Csize, data = gdf, iter = 999, turbo = TRUE,
RRPP = TRUE, print.progress = FALSE)
summary(fit)

### Extracting objects and plotting options
# diagnostic plots
```

```
plot(fit, type = "diagnostics")
# diagnostic plots, including plotOutliers
plot(fit, type = "diagnostics", outliers = TRUE)

# PC plot rotated to major axis of fitted values
plot(fit, type = "PC", pch = 19, col = "blue")
# Regression-type plots

# Use fitted values from the model to make prediction lines
plot(fit, type = "regression",
predictor = gdf$Csize, reg.type = "RegScore",
pch = 19, col = "green")

# Uses coefficients from the model to find the projected regression
# scores
rat.plot <- plot(fit, type = "regression",
predictor = gdf$Csize, reg.type = "RegScore",
pch = 21, bg = "yellow")

# TPS grids for min and max scores in previous plot
preds <- shape.predictor(fit$GM$fitted, x = rat.plot$RegScore,
                         predmin = min(rat.plot$RegScore),
                         predmax = max(rat.plot$RegScore))
M <- rat.gpa$consensus
plotRefToTarget(M, preds$predmin, mag=2)
plotRefToTarget(M, preds$predmax, mag=2)

attributes(fit)
fit$fitted[1:3, ] # the fitted values (first three specimens)
fit$GM$fitted[,, 1:3] # the fitted values as Procrustes
# coordinates (same specimens)

### THE FOLLOWING ILLUSTRATES SIMPLER SOLUTIONS FOR
### THE NOW DEFUNCT nested.update FUNCTION USING
### THE anova GENERIC FUNCTION

data("larvalMorph")
Y.gpa <- gpagen(larvalMorph$tailcoords,
curves = larvalMorph$tail.sliders,
ProcD = TRUE, print.progress = FALSE)
gdf <- geomorph.data.frame(Y.gpa, treatment = larvalMorph$treatment,
family = larvalMorph$family)

fit <- procD.lm(coords ~ treatment/family, data = gdf, turbo = TRUE,
print.progress = FALSE, iter = 999)
anova(fit) # treatment effect not adjusted
anova(fit, error = c("treatment:family", "Residuals"))
# treatment effect updated (adjusted)

## End(Not run)
```

procD.pgls *Phylogenetic ANOVA/regression for Procrustes shape variables*

Description

Function performs Procrustes ANOVA in a phylogenetic framework and uses permutation procedures to assess statistical hypotheses describing patterns of shape variation and covariation for a set of Procrustes-aligned coordinates

Usage

```
procD.pgls(
  f1,
  phy,
  Cov = NULL,
  iter = 999,
  seed = NULL,
  int.first = FALSE,
  SS.type = c("I", "II", "III"),
  effect.type = c("F", "cohen"),
  data = NULL,
  print.progress = TRUE,
  ...
)
```

Arguments

f1	A formula for the linear model (e.g., y ~ x1 + x2)
phy	A phylogenetic tree of class phylo - see read.tree in library ape
Cov	An optional covariance matrix that can be used for generalized least squares estimates of coefficients and sums of squares and cross-products (see Adams and Collyer 2018), if one wishes to override the calculation of a covariance matrix based on a Brownian Motion model of evolution. Using this argument essentially turns this function into an alternate version of procD.lm.
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
int.first	A logical value to indicate if interactions of first main effects should precede subsequent main effects
SS.type	SS.type A choice between type I (sequential), type II (hierarchical), or type III (marginal) sums of squares and cross-products computations.

effect.type	One of "F" or "cohen", to choose from which random distribution to estimate effect size. (The default is "F". The option, "cohen", refers to Cohen's f-squared values. Values are log-transformed before z-score calculation to assure normally distributed effect sizes.)
data	A data frame for the function environment, see geomorph.data.frame
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.
...	Arguments passed on to procD.lm.

Details

The function performs ANOVA and regression models in a phylogenetic context under a Brownian motion model of evolution, in a manner that can accommodate high-dimensional datasets. The approach is derived from the statistical equivalency between parametric methods utilizing covariance matrices and methods based on distance matrices (Adams 2014). Data input is specified by a formula (e.g., y ~ X), where 'y' specifies the response variables (shape data), and 'X' contains one or more independent variables (discrete or continuous). The response matrix 'Y' can be either in the form of a two-dimensional data matrix of dimension (n x [p x k]), or a 3D array (p x n x k). It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. Linear model fits (using the lm function) can also be input in place of a formula. Arguments for lm can also be passed on via this function. The user must also specify a phylogeny describing the evolutionary relationships among species (of class phylo). Note that the specimen labels for both X and Y must match the labels on the tips of the phylogeny.

The function two.d.array can be used to obtain a two-dimensional data matrix from a 3D array of landmark coordinates; however this step is no longer necessary, as procD.lm can receive 3D arrays as dependent variables. It is also recommended that geomorph.data.frame is used to create and input a data frame. This will reduce problems caused by conflicts between the global and function environments. In the absence of a specified data frame, procD.pgls will attempt to coerce input data into a data frame, but success is not guaranteed.

From the phylogeny, a phylogenetic transformation matrix is obtained under a Brownian motion model, and used to transform the X and Y variables. Next, the Gower-centered distance matrix is obtained from predicted values from the model (Y ~ X), from which sums-of-squares, F-ratios, and R-squared are estimated for each factor in the model (see Adams, 2014). Data are then permuted across the tips of the phylogeny, and all estimates of statistical values are obtained for the permuted data, which are compared to the observed value to assess significance. This approach has been shown to have appropriate type I error rates (Adams and Collyer 2018), whereas an alternative procedure for phylogenetic regression of morphometric shape data displays elevated type I error rates (see Adams and Collyer 2015).

Effect-sizes (Z scores) are computed as standard deviates of either the F or Cohen's f-squared sampling distributions generated, which might be more intuitive for P-values than F-values (see Collyer et al. 2015). Values from these distributions are log-transformed prior to effect size estimation, to assure normally distributed data. The SS type will influence how Cohen's f-squared values are calculated. Cohen's f-squared values are based on partial eta-squared values that can be calculated sequentially or marginally, as with SS.

In the case that multiple factor or factor-covariate interactions are used in the model formula, one can specify whether all main effects should be added to the model first, or interactions should

precede subsequent main effects (i.e., Y ~ a + b + c + a:b + ..., or Y ~ a + b + a:b + c + ..., respectively.)

The generic functions, print, summary, and plot all work with procD.pgls. The generic function, plot, produces diagnostic plots for Procrustes residuals of the linear fit.

Notes for geomorph 3.0.6 and subsequent versions:

Compared to previous versions, GLS computations in random permutations require RRPP (Adams and Collyer 2018). Thus, full randomization is no longer permitted with this function. This function uses procD.lm, after calculating a phylogenetic covariance matrix, and with the constraint of RRPP. If alternative covariance matrices or permutation methods are preferred, one can use procD.lm, which has greater flexibility.

Notes for geomorph 3.0.4 and subsequent versions:

Compared to previous versions of geomorph, users might notice differences in effect sizes. Previous versions used z-scores calculated with expected values of statistics from null hypotheses (sensu Collyer et al. 2015); however Adams and Collyer (2016) showed that expected values for some statistics can vary with sample size and variable number, and recommended finding the expected value, empirically, as the mean from the set of random outcomes. Geomorph 3.0.4 and subsequent versions now center z-scores on their empirically estimated expected values and where appropriate, log-transform values to assure statistics are normally distributed. This can result in negative effect sizes, when statistics are smaller than expected compared to the average random outcome. For ANOVA-based functions, the option to choose among different statistics to measure effect size is now a function argument.

Value

procD.lm.pgls returns an object of class "procD.lm". See procD.lm for a description of the list of results generated. Additionally, procD.pgls provides the phylogenetic correction matrix, Pcor, plus "pgls" adjusted coefficients, fitted values, residuals, and mean.

Author(s)

Dean Adams and Michael Collyer

References

Adams, D.C. 2014. A method for assessing phylogenetic least squares models for shape and other high-dimensional multivariate data. Evolution. 68:2675-2688.

Adams, D.C., and M.L. Collyer. 2015. Permutation tests for phylogenetic comparative analyses of high-dimensional shape data: what you shuffle matters. Evolution. 69:823-829.

Collyer, M.L., D.J. Sekora, and D.C. Adams. 2015. A method for analysis of phenotypic change for phenotypes described by high-dimensional data. Heredity. 115:357-365.

Adams, D.C. and M.L. Collyer. 2016. On the comparison of the strength of morphological integration across morphometric datasets. Evolution. 70:2623-2631.

Adams, D.C. and M.L. Collyer. 2018. Multivariate comparative methods: evaluations, comparisons, and recommendations. Systematic Biology. 67:14-31.

Examples

```
### Example of D-PGLS for high-dimensional data
data(plethspecies)
Y.gpa<-gpagen(plethspecies$land)    #GPA-alignment
gdf <- geomorph.data.frame(Y.gpa, phy = plethspecies$phy)
pleth.pgls <- procD.pgls(coords ~ Csize, phy = phy, data = gdf,
iter = 999)
anova(pleth.pgls)
summary(pleth.pgls)  #similar output

### Working with procD.pgls objects
predict(pleth.pgls)
plot(pleth.pgls, type="regression", reg.type="RegScore",
predictor = gdf$Csize)
attributes(pleth.pgls) # Note the PGLS object
attributes(pleth.pgls$PGLS) # PGLS details embedded within PGLS object
pleth.pgls$LM$Pcov # the projection matrix derived from the
# phylogenetic covariance matrix
pleth.pgls$pgls.fitted # the PGLS fitted values
pleth.pgls$GM$pgls.fitted # The same fitted values, in a 3D array
```

---

pupfish *Landmarks on pupfish*

Description

Landmark data from Cyprindon pecosensis body shapes, with indication of Sex and Population from which fish were sampled (Marsh or Sinkhole).

Details

These data were previously aligned with GPA. Centroid size (CS) is also provided.

Author(s)

Michael Collyer

References

Collyer, M.L., D.J. Sekora, and D.C. Adams. 2015. A method for analysis of phenotypic change for phenotypes described by high-dimensional data. Heredity. 115: 357-365.

ratland *Landmark data from dataset rat*

Description

Landmark data from dataset rat

Author(s)

Dean Adams

References

Bookstein, F. L. 1991. Morphometric tools for landmark data: Geometry and Biology. Cambridge Univ. Press, New York.

read.morphologika *Read landmark data from Morphologika file(s)*

Description

Read Morphologika file (*.txt) to obtain landmark coordinates and specimen information

Usage

```
read.morphologika(filelist)
```

Arguments

filelist	The name of a Morphologika *.txt file containing two- or three-dimensional landmark data. Alternatively, a character vector of names of Morphologika *.txt file (such as made using list.files)

Details

This function reads a *.txt file in the Morphologika format containing two- or three-dimensional landmark coordinates. Morphologika files are text files in one of the standard formats for geometric morphometrics (see O'Higgins and Jones 1998,2006). If multiple morphologika files are specified (containing the same number of landmarks for all specimens), function returns a single object for files.

If the headers "[labels]", "[labelvalues]" and "[groups]" are present, then a data matrix containing all individual specimen information is returned. If the header "[wireframe]" is present, then a matrix of the landmark addresses for the wireframe is returned (see plotRefToTarget option 'links'). If the header "[polygon]" is present, then a matrix of the landmark addresses for the polygon wireframe is returned (see polygon3d or polygon).

NOTE: For multiple morphologika files that each contain only a single specimen (such as those exported from Stratovan Checkpoint software), one can add specimen names to the returned 3D array by: dimnames(mydata)[[3]] <- gsub (".txt", "", filelist)).

Value

Function returns a (p x k x n) array of the coordinate data. If other optional headers are present in the file (e.g. "[labels]" or "[wireframe]") function returns a list containing the "coords" array, and data matrix of "labels" and or "wireframe".

Author(s)

Emma Sherratt & Erik Otarola-Castillo

References

O'Higgins P and Jones N (1998) Facial growth in Cercocebus torquatus: An application of three dimensional geometric morphometric techniques to the study of morphological variation. Journal of Anatomy. 193: 251-272

O'Higgins P and Jones N (2006) Tools for statistical shape analysis. Hull York Medical School.

---

read.ply *Read mesh data (vertices and faces) from ply files*

Description

A function to read ply files, which can be used for digitizing landmark coordinates or for shape warps.

Usage

```
read.ply(file, ShowSpecimen = TRUE, addNormals = TRUE)
```

Arguments

file	An ASCII ply file
ShowSpecimen	logical Indicating whether or not the ply file should be displayed
addNormals	logical Indicating whether or not the normals of each vertex should be calculated (using addNormals)

Details

Function reads three-dimensional surface data in the form of a single ply file (Polygon File Format; ASCII format only, from 3D scanners such as NextEngine and David scanners). Vertices of the surface may then be used to digitize three-dimensional points, and semilandmarks on curves and surfaces. The surface may also be used as a mesh for visualizing 3D deformations (warpRefMesh). The function opens the ply file and plots the mesh, with faces rendered if file contains faces, and colored if the file contains vertex color. Vertex normals allow better visualization and more accurate digitizing with digit.fixed.

Value

Function returns the following components:

mesh3d	list of class mesh3d- see rgl for details

Author(s)

Dean Adams & Emma Sherratt

See Also

rgl-package (used in 3D plotting)

Examples

```
# If the file has no mesh color, or color is undesirable, user can
# assign this as follows:
# Using the example scallop PLY
data(scallopPLY)
myply <- scallopPLY$ply
myply$material$color <- "gray" # using color word
myply$material$color <- "#FCE6C9" # using RGB code
```

---

readland.fcsv *Read landmark data matrix from fcsv file*

Description

Read landmark data for a single specimen from an fcsv file obtained from SlicerMorph.

Usage

```
readland.fcsv(file = NULL)
```

Arguments

file	the name of a *.fcsv file containing three-dimensional landmark data to be read in

Details

This function merely extracts x, y, z coordinates from an fcsv file.

Value

Function returns a p x 3 matrix of x, y, z coordinates for p landmarks.

Author(s)

Murat Maga and Michael Collyer

`readland.nts` *Read landmark data matrix from nts file*

Description

Read single *.nts file containing landmark coordinates for one or more specimens

Usage

```
readland.nts(file)
```

Arguments

`file`	the name of a *.nts file containing two- or three-dimensional landmark data to be read in

Details

Function reads a single *.nts file containing two- or three-dimensional landmark coordinates.

NTS files are text files in one of the standard formats for geometric morphometrics (see Rohlf 2012). Multiple specimen format: The parameter line contains 5 or 6 elements, and must begin with a "1" to designate a rectangular matrix. The second and third values designate how many specimens (n) and how many total variables (p x k) are in the data matrix. The fourth value is a "0" if the data matrix is complete and a "1" if there are missing values. If missing values are present, the '1' is followed by the arbitrary numeric code used to represent missing values (e.g., -999). These values will be replaced with "NA" in the output array. Subsequent analyses requires a full complement of data, see `estimate.missing`. The final value of the parameter line denotes the dimensionality of the landmarks (2,3) and begins with "DIM=". If specimen and variable labels are included, these are designated placing an "L" immediately following the specimen or variable values in the parameter file. The labels then precede the data matrix.

Missing data may also be represented by designating them using 'NA'. In this case, the standard NTSYS header is used with no numeric designation for missing data (i.e. the fourth value is '0'). The positions of missing landmarks may then be estimated using estimate.missing.

Special NTS files: *.dta files in the written by IDAV Landmark Editor, and *.nts files written by Stratovan Checkpoint have incorrect header notation; every header is 1 n p-x-k 1 9999 Dim=3, rather than 1 n p-x-k 0 Dim=3, which denotes that missing data is in the file even when it is not. Users must change manually the header (in a text editor) before using this function

Value

Function returns a 3D array (p x k x n), where p is the number of landmark points, k is the number of landmark dimensions (2 or 3), and n is the number of specimens. The third dimension of this array contains names for each specimen, which are obtained from the names in the *.nts file (if included).

Author(s)

Dean Adams & Emma Sherratt

References

Rohlf, F. J. 2012 NTSYSpc: Numerical taxonomy and multivariate analysis system. Version 2.2. Exeter Software, New York.

See Also

readmulti.nts

---

readland.shapes *Read landmark data from a shapes object (StereoMorph)*

Description

Read data from an object with class shapes, created by StereoMorph

Usage

```
readland.shapes(
  Shapes,
  nCurvePts = NULL,
  continuous.curve = NULL,
  scaled = TRUE
)
```

Arguments

Shapes	An object with class, "shapes"
nCurvePts	A single value (if only one curve) or a string of values (if multiple curves) for culling the number of curve points, using linear interpolation to find equally spaced distances between points. For example, with three curves, nCurvePts = c(20, 50, 10) would return 20, 50, and 10 curve points, respectively. If fixed landmarks are end points of curves, they will not be repeated as additional curve points. Therefore, one should expect in these cases that the number of semilandmarks are fewer than the number of curve points; i.e., choosing 20 curve points might yield 18 semilandmarks. If left NULL, no curve points will be estimated and only fixed landmarks will be read. Note: if less than 3 curve points are chosen for an open curve or less than 4 points are chosen for a closed curve, the number of points will default to 0, as fewer than these make it impossible to develop a curves matrix. A value of 0 can be chosen to omit specific curves when reading in data.

`continuous.curve`
: An optional value or string of values to indicate which curves are closed and whose same start and end point should be treated as a semilandmark. For example, if one wishes to digitize a curve around an eye, for digitizing purposes a point might be initiated to find a curve around the eye and back to the point. This point will be "fixed" without indicating the curve is continuous. One could use this argument to indicate which curves are continuous; e.g., continuous.curve = c(2, 5) to indicate curves 2 and 5 have starting points that are not fixed landmarks but really sliding semilandmarks, as part of a closed curve.

`scaled`
: A logical value (TRUE as default) to indicate whether scaled landmarks and curve points should be used. If any scales are missing, the function will default to scaled = FALSE.

Details

This function reads data from an object with class "shapes", after digitizing specimens using StereoMorph. This function can read in landmarks plus acquire points from curves, if the curves option was used in digitizing. This function is intended for reading data and facilitating the generation of a "curves" matrix for gpagen. It is not intended to influence how one should digitize landmarks using StereoMorph. Each digitizing experience is unique and users might need to edit their own data using StereoMorph functions in some cases; this function will not necessarily overcome all data editing challenges. This function currently works with 2D data only, as the readShapes function of StereoMorph pertains to digitizing images.

The enhanced feature of this function is that it can find a prescribed number of approximately equally spaced semilandmarks along 2D curves (from the many points of Bezier curves in StereoMorph), facilitating rapid gpagen analysis and the flexibility to change the desired number of semilandmarks defining curves. This is only true, however, if the curves option in digitizeImages (StereoMorph) is used.

The user can specify the number of points along curves, whether curves are continuous (closed and without fixed points), and whether landmarks should be scaled, if possible (if scaling was performed while digitizing).

Value

An object of class "geomorphShapes" is a list containing the following

`landmarks`
: A list of specimen by specimen landmarks, arranged with fixed landmarks first and semilandmarks second.

`fixed`
: A vector indicating which landmarks are initially considered "fixed" (but this can be changed).

`sliders`
: A vector indicating which landmarks are initially considered semilandmarks, or "sliders".

`curves`
: A matrix used in gpagen to define how semilandmarks slide by tangents described by flanking points. Each row of the matrix is a series of three points: tangent point 1, slider, tangent point 2. This matrix can be edited. to remove sliding points or add some, if points were originally considered fixed. This matrix is merely a suggestion, based on the curve information read in from a shapes object. Users should be able to rearrange this matrix, as needed.

n	The number of specimens.
p	The number of (both fixed and semi-) landmarks.
k	The number of landmark dimensions, currently only 2.
scaled	Logical value to indicate in landmarks are scaled.

Author(s)

Michael Collyer

See Also

readShapes (from StereoMorph)

Examples

```
# A true example is not possible, as digitizing experiences are
# unique, but here is a general set-up
# myShapes <- readShapes("myDigitizingFile") # data from readShapes
# from StereoMorph
# myGMdata <- readland.shapes(myShapes) # just reading in the fixed
# landmarks
# myGMdata <- readland.shapes(myShapes,
#      nCurvePts = c(10, 15, 5),
#      continuous.curve = 2) # fixed landmarks plus curve points
# for three curves, one closed
# myGPA <- gpagen(myGMdata, ProcD = FALSE) # GPA perfomed with
# minimized bending energy
```

---

readland.tps *Read landmark data from tps file*

Description

Read *.tps file to obtain landmark coordinates

Usage

```
readland.tps(
  file,
  specID = c("None", "ID", "imageID"),
  negNA = FALSE,
  readcurves = FALSE,
  warnmsg = TRUE
)
```

Arguments

file	A *.tps file containing two- or three-dimensional landmark data
specID	a character specifying whether to extract the specimen ID names from the ID or IMAGE lines (default is "None").
negNA	A logical value indicating whether negative landmark coordinates in the tps file should be imported as missing values and coded as 'NA' (TRUE), or imported as such.
readcurves	A logical value stating whether CURVES= field and associated coordinate data will be read as semilandmarks (TRUE) or ignored (FALSE).
warnmsg	A logical value stating whether warnings should be printed

Details

This function reads a *.tps file containing two- or three-dimensional landmark coordinates. Tps files are text files in one of the standard formats for geometric morphometrics (see Rohlf 2010). Two-dimensional landmarks coordinates are designated by the identifier "LM=", while three-dimensional data are designated by "LM3=". Landmark coordinates are multiplied by their scale factor if this is provided for all specimens. If one or more specimens are missing the scale factor, landmarks are treated in their original units.

Missing data may be present in the file. If data were digitized in geomorph or StereoMorph, these are automatically identified. If, instead, data digitizing took place in a software package that records missing values as negative coordinates (e.g. tpsDig), the user needs to specify whether negative values should be transformed to 'NAs' through the argument neg.NA = TRUE. The positions of missing landmarks may then be estimated using `estimate.missing`.

The user may specify whether specimen names are to be extracted from the 'ID=' field or 'IMAGE=' field and included in the resulting 3D array. e.g., for 'ID=' use (file, specID = "ID") and for 'IMAGE=' use (file, specID = "imageID"). The default is specID="None".

If there are curves defined in the file (i.e., CURVES= fields), the option 'readcurves' should be used. When readcurves = TRUE, the coordinate data for the curves will be returned as semilandmarks and will be appended to the fixed landmark data. Then the user needs to use `define.sliders` or `define.sliders` to create a matrix designating how the curve points will slide (used by 'curves=' in `gpagen`). When readcurves = FALSE, only the landmark data are returned.

NOTE: At present, all other information that can be contained in tps files (comments, variables, radii, etc.) is ignored.

Value

Function returns a (p x k x n) array, where p is the number of landmark points, k is the number of landmark dimensions (2 or 3), and n is the number of specimens. The third dimension of this array contains names for each specimen, which are obtained from the image names in the *.tps file.

Author(s)

Dean Adams, Emma Sherratt, Michael Collyer & Antigoni Kaliontzopoulou

References

Rohlf, F. J. 2010. tpsRelw: Relative warps analysis. Version 1.49. Department of Ecology and Evolution, State University of New York at Stony Brook, Stony Brook, NY.

`readmulti.nts` *Read and combine multiple nts files*

Description

Read multiple nts (or .dta) files to obtain landmark coordinates and combine them into a single array

Usage

```
readmulti.nts(filelist)
```

Arguments

`filelist`	A vector containing the file paths to all the nts files to be compiled

Details

This is a wrapper of `readland.nts` to allow reading landmark coordinates, in 2D or 3D, from several nts (or .dta) files, and compiling them into an array for proceeding with GM procedures.

See `readland.nts` for adequately formatting NTS files and requirements that need to be met for that (and therefore this) function to work correctly.

Value

Function returns a 3D array (p x k x n), where p is the number of landmark points, k is the number of landmark dimensions (2 or 3), and n is the number of specimens. The third dimension of this array contains names for each specimen, which are retrieved from the nts specimen labels, if those are available. Specimens in nts files without labels are named as filename_# where # are consecutive numbers.

Author(s)

Antigoni Kaliontzopoulou

References

Rohlf, F. J. 2012 NTSYSpc: Numerical taxonomy and multivariate analysis system. Version 2.2. Exeter Software, New York.

readmulti.tps *Read and combine multiple tps files*

Description

Read multiple tps files to obtain landmark coordinates and combine them into a single array

Usage

```
readmulti.tps(filelist, ...)
```

Arguments

filelist	A vector containing the file paths to all the tps files to be compiled
...	other arguments to be passed to readland.tps

Details

This is a wrapper of readland.tps to allow reading landmark coordinates, in 2D or 3D, from several tps files , and compiling them into an array for proceeding with GM procedures.

The arguments specID and negNA of readland.tps can be directly set through this function. Note that if specID is set to either "None" or "ID", a check for duplicate specimen names is not possible and specimens will be numbered with 1:N, where N the total number of specimens.

Value

Function returns a (p x k x n) array, where p is the number of landmark points, k is the number of landmark dimensions (2 or 3), and n is the total number of specimens across all tps files included in the folder read.

Author(s)

Antigoni Kaliontzopoulou

rotate.coords *Rotate or flip landmark or coordinate configurations*

Description

Function to rotate or flip 2D landmark or coordinate configurations to re-orientate them, as desired.

Usage

```
rotate.coords(
  A,
  type = c("flipX", "flipY", "rotateC", "rotateCC"),
  index = NULL
)
```

Arguments

A	One of either an array of landmark coordinates (p, 2, n dimensions for n specimens and p 2D points), a class gpagen object, or a p x 2 matrix for a single specimen.
type	The type of rotation or flip to be performed. Specimens can be flipped with respect to x or y axes, or rotated clockwise (C) or counter-clockwise (CC).
index	An index to indicate which specimens should be rotated or flipped. If NULL (default) all specimens are rotated. A binary index (0 = do not rotate; 1 = rotate) as a vector with the same length as the number of specimens will direct which specimens are manipulated. A factor with two levels or a numeric vector with two any two values can also be used. The second level (or larger value) will be the level manipulated. For example, a factor with levels = c("a", "b") will rotate specimens matching level "b". This function might be useful for reflecting specimens with bilateral structures.

Details

This function will allow a user to rotate or flip one or several configurations of raw landmarks or Procrustes residuals from GPA, as desired. This function only works with two-dimensional configurations. This is a useful tool if importing coordinates or performing GPA produces undesired orientations (such as flipping configurations upside down, when aligning them to their PCs). It is not as useful with 3D coordinates, as the plotting tools for 3D coordinates already have built-in rotation capabilities.

The function returns an object of the same class as input, after having rotated or flipped the coordinates of each specimen. If multiple steps are required, the function can be used in a recursive fashion.

Author(s)

Michael Collyer

Examples

```
data(plethodon)
Y.gpa <- gpagen(plethodon$land)
plot(Y.gpa)
Y.gpa2 <- rotate.coords(Y.gpa, "flipX")
plot(Y.gpa2)
Y.gpa3 <- rotate.coords(Y.gpa2, "rotateCC")
plot(Y.gpa3)
```

```
spec1 <- Y.gpa$coords[,,1]
plot(spec1, asp = 1)
spec1 <- rotate.coords(spec1, "flipY")
plot(spec1, asp = 1)

specs1to3 <- Y.gpa$coords[,,1:3]
plotAllSpecimens(specs1to3)
specs1to3 <- rotate.coords(specs1to3, "rotateC")
plotAllSpecimens(specs1to3)
```

`scallopPLY` *3D scan of a scallop shell from a .ply file in mesh3d format*

Description

3D scan of a scallop shell from a .ply file in mesh3d format

Author(s)

Emma Sherratt

References

Serb et al. (2011). "Morphological convergence of shell shape in distantly related scallop species (Mollusca: Pectinidae)." Zoological Journal of the Linnean Society 163: 571-584.

`scallops` *Landmark data from scallop shells*

Description

Landmark data from scallop shells

Author(s)

Dean Adams and Erik Otarola-Castillo

References

Serb et al. (2011). "Morphological convergence of shell shape in distantly related scallop species (Mollusca: Pectinidae)." Zoological Journal of the Linnean Society 163: 571-584.

shape.predictor *Shape prediction from numeric predictors*

Description

Function estimates one or more configurations based on one or more linear predictors, such as PC scores allometric relationships, or any other least squares or partial least squares regression. These configurations can be used with plotRefToTarget to generate graphical representations of shape change, based on prediction criteria.

Usage

```
shape.predictor(A, x = NULL, Intercept = FALSE, method = c("LS", "PLS"), ...)
```

Arguments

A	A 3D array (p x k x n) containing Procrustes shape variables either from GPA or fitted values from a previous analytical procedure.
x	Linear (numeric) predictors. Can be a vector or a matrix, or a list containing vectors or matrices. Values must be numeric. If a factor is desired, one should use model.matrix to obtain a design matrix. This will impact how prediction criteria need to be provided (see below).
Intercept	Logical value to indicate whether an intercept should be used in the linear equation for predictions. Generally, this value will be FALSE for shape predictions made in ordination plots. It should be TRUE in cases where the expected shape at the point the predictor has a value of 0 is not the mean shape.
method	A choice between least squares (LS) or partial least squares (PLS) regression for prediction. The function defaults to LS prediction. PLS might be chosen in cases where correlation is preferred over linear regression. If PLS is chosen, a two-block PLS analysis using two.b.pls should be performed first, as only the first singular vector for predictors will be used for defining prediction criteria (see below).
...	Any number of prediction criteria. Criteria should be presented as either a scalar (if one predictor is provided) or a vector (if more than one predictor or a prediction matrix is provided); e.g., pred1 = c(0.1, -0.5), pred2 = c(-0.2, -0.1) (which would be the case if two predictors were provided). It is essential that the number of elements in any prediction criterion matches the number of predictors. Caution should be used when providing a design matrix to ensure that correct dummy variables are used in prediction criteria, and that either 1) an intercept is not included in the design and 2) is TRUE in the Intercept argument; or or 1) an intercept is included in the design and 2) is FALSE in the Intercept argument; or 1) an intercept is not included in the design and 2) is FALSE in the Intercept argument, if no intercept is desired.

Value

A list of predicted shapes matching the number of vectors of prediction criteria provides. The predictions also have names matching those of the prediction criteria.

Author(s)

Michael Collyer

Examples

```
# Examples using Plethodon data

data("plethodon")

Y.gpa <- gpagen(plethodon$land)    #GPA-alignment
plot(gm.prcomp(Y.gpa$coords))

preds <- shape.predictor(Y.gpa$coords, x= NULL, Intercept = FALSE,
pred1 = -0.1, pred2 = 0.1) # PC 1 extremes, sort of
M <- mshape(Y.gpa$coords)
plotRefToTarget(M, preds$pred1)
plotRefToTarget(M, preds[[1]]) # same result
plotRefToTarget(M, preds$pred2)

PCA <- gm.prcomp(Y.gpa$coords)
PC <- PCA$x[,1]
preds <- shape.predictor(Y.gpa$coords, x= PC, Intercept = FALSE,
pred1 = min(PC), pred2 = max(PC)) # PC 1 extremes, more technically
plotRefToTarget(M, preds$pred1)
plotRefToTarget(M, preds$pred2)

PC <- PCA$x[,1:2]
# user-picked spots can be anything, but it in this case, apparent groups
preds <- shape.predictor(Y.gpa$coords, x= PC, Intercept = FALSE,
                         pred1 = c(0.045,-0.02),
                         pred2 = c(-0.025,0.06),
                         pred3 = c(-0.06,-0.04))
plotRefToTarget(M, preds$pred1)
plotRefToTarget(M, preds$pred2)
plotRefToTarget(M, preds$pred3)

# allometry example - straight-up allometry

preds <- shape.predictor(Y.gpa$coords, x= log(Y.gpa$Csize),
                         Intercept = TRUE,
                         predmin = min(log(Y.gpa$Csize)),
                         predmax = max(log(Y.gpa$Csize)))

plotRefToTarget(M, preds$predmin, mag=3)
plotRefToTarget(M, preds$predmax, mag=3)

# allometry example - using RegScore or PredLine via procD.lm
```

```
gdf <- geomorph.data.frame(Y.gpa)
plethAllometry <- procD.lm(coords ~ log(Csize), data=gdf)
allom.plot <- plot(plethAllometry,
type = "regression",
predictor = log(gdf$Csize),
reg.type ="RegScore") # make sure to have a predictor

preds <- shape.predictor(plethAllometry$GM$fitted,
                         x= allom.plot$RegScore, Intercept = FALSE,
                         predmin = min(allom.plot$RegScore),
                         predmax = max(allom.plot$RegScore))
plotRefToTarget(M, preds$predmin, mag=3)
plotRefToTarget(M, preds$predmax, mag=3)

allom.plot <- plot(plethAllometry,
type = "regression",
predictor = log(gdf$Csize),
reg.type ="PredLine")
preds <- shape.predictor(plethAllometry$GM$fitted,
                         x= allom.plot$PredLine, Intercept = FALSE,
                         predmin = min(allom.plot$PredLine),
                         predmax = max(allom.plot$PredLine))
plotRefToTarget(M, preds$predmin, mag=3)
plotRefToTarget(M, preds$predmax, mag=3)

# using factors via PCA

gdf <- geomorph.data.frame(Y.gpa, species = plethodon$species,
       site = plethodon$site)
pleth <- procD.lm(coords ~ species*site, data=gdf)
PCA <- prcomp(pleth$fitted)
plot(PCA$x, asp=1, pch=19)

means <- unique(round(PCA$x,3))
means # note: suggests 3 PCs useful enough

preds <- shape.predictor(arrayspecs(pleth$fitted, 12,2), x= PCA$x[,1:3],
                         Intercept = FALSE,
                         pred1 = means[1,1:3],
                         pred2 = means[2,1:3],
                         pred3 = means[3,1:3],
                         pred4 = means[4,1:3])
plotRefToTarget(M, preds$pred1, mag=2)
plotRefToTarget(M, preds$pred2, mag=2)
plotRefToTarget(M, preds$pred3, mag=2)
plotRefToTarget(M, preds$pred4, mag=2)

# Using a design matrix for factors

X <- pleth$X
X # includes intercept; remove for better functioning
X <- X[,-1]
```

```
symJord <- c(0,1,0) # design for P. Jordani in sympatry
alloJord <- c(0,0,0) # design for P. Jordani in allopatry
preds <- shape.predictor(arrayspecs(pleth$fitted, 12,2), x = X,
                         Intercept = TRUE,
                         symJord=symJord, alloJord=alloJord)
plotRefToTarget(M, preds$symJord, mag=2)
plotRefToTarget(M, preds$alloJord, mag=2)

# PLS Example

data(plethShapeFood)
Y.gpa<-gpagen(plethShapeFood$land)    #GPA-alignment

# 2B-PLS between head shape and food use data
PLS <-two.b.pls(A1 = plethShapeFood$food, A2 = Y.gpa$coords, iter=999)
summary(PLS)
plot(PLS)

preds <- shape.predictor(Y.gpa$coords, plethShapeFood$food,
                         Intercept = FALSE,
                         method = "PLS",
                         pred1 = 2, pred2 = -4, pred3 = 2.5)
                         # using PLS plot as a guide
M <- mshape(Y.gpa$coords)
plotRefToTarget(M, preds$pred1, mag=2)
plotRefToTarget(M, preds$pred2, mag=2)
plotRefToTarget(M, preds$pred3, mag=2)
```

---

shapeHulls *Update Plots with Convex Hulls for Groups*

Description

This function is used to update plot.procD.lm and plot.gm.prcomp ordination plot objects with convex hulls for different groups. If no groups are defined (groups is NULL) just a single convex hull will be returned. Groups do not need to be a factor in the original procD.lm fit.

Usage

```
shapeHulls(
  x,
  groups = NULL,
  group.cols = NULL,
  group.lwd = NULL,
  group.lty = NULL
)
```

Arguments

x	A plot.procD.lm or plot.gm.prcomp plot object.
groups	An optional vector or factor to define groups for hull. If NULL, only one hull will be generated for all points.
group.cols	An optional vector to define hull colors, arranged in the same order as factor levels. If NULL and if multiple groups exist, the general R color sequence (black, red, green, blue, etc.) will be used.
group.lwd	An optional vector equal in length to the number of group levels, and arranged in the order of group levels, to modify hull line width.
group.lty	An optional vector equal in length to the number of group levels, and arranged in the order of group levels, to modify hull line type.

Details

This function is a wrapper for the points function. It is intentionally limited, so as to not interfere with other plot parameter adjustments.

Author(s)

Michael Collyer

See Also

procD.lm

Examples

```
# Via procD.lm and plot.procD.lm

data("pupfish")
gdf <- geomorph.data.frame(coords = pupfish$coords, Sex = pupfish$Sex,
Pop = pupfish$Pop)
fit <- procD.lm(coords ~ Pop * Sex, data = gdf, print.progress = FALSE)
pc.plot <- plot(fit, type = "PC", pch = 19)
shapeHulls(pc.plot)

pc.plot <- plot(fit, type = "PC", pch = 19)
groups <- interaction(gdf$Pop, gdf$Sex)

shapeHulls(pc.plot, groups = groups,
group.cols = c("dark red", "dark red", "dark blue", "dark blue"),
group.lwd = rep(2, 4), group.lty = c(2, 1, 2, 1))

legend("topright", levels(groups),
col = c("dark red", "dark red", "dark blue", "dark blue"),
lwd = rep(2,4), lty = c(2, 1, 2, 1))

pc.plot <- plot(fit, type = "PC", pch = 19)
```

```
shapeHulls(pc.plot, groups = gdf$Sex, group.cols = c("black", "black"),
group.lwd = rep(2, 2), group.lty = c(2, 1))
legend("topright", levels(gdf$Sex), lwd = 2, lty = c(2, 1))

# Via gm.prcomp and plot.gm.prcomp

data(plethspecies)
Y.gpa <- gpagen(plethspecies$land)    #GPA-alignment
pleth.phylo <- gm.prcomp(Y.gpa$coords, phy = plethspecies$phy)
summary(pleth.phylo)

pc.plot <- plot(pleth.phylo, phylo = TRUE)
gp <- factor(c(rep(1, 5), rep(2, 4)))
shapeHulls(pc.plot, groups = gp, group.cols = 1:2,
group.lwd = rep(2, 2), group.lty = c(2, 1))
legend("topright", c("P. cinereus clade", "P. hubrichti clade"),
col = 1:2, lwd = 2, lty = c(2, 1))
```

`summary.bilat.symmetry` *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'bilat.symmetry'
summary(object, ...)
```

Arguments

`object`	print/summary object (from `bilat.symmetry`)
`...`	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.combined.set *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'combined.set'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.compare.CR *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'compare.CR'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.compare.pls *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'compare.pls'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.CR *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'CR'
summary(object, ...)
```

Arguments

object	print/summary object (from phylo.modularity)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.CR.phylo *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'CR.phylo'
summary(object, ...)
```

Arguments

object	print/summary object (from phylo.modularity)
...	other arguments passed to print/summary

Author(s)

Dean Adams

summary.evolrate *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'evolrate'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.evolrate1 *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'evolrate1'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.geomorphShapes *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'geomorphShapes'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.gm.prcomp *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'gm.prcomp'
summary(object, ...)
```

Arguments

object	print/summary object
...	other arguments passed to print/summary

Author(s)

Antigoni Kaliontzopoulou

summary.gpagen *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'gpagen'
summary(object, ...)
```

Arguments

object	print/summary object (from gpagen)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.morphol.disparity *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'morphol.disparity'
summary(object, ...)
```

Arguments

object	print/summary object (from morphol.disparity)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.physignal *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'physignal'
summary(object, ...)
```

Arguments

object	print/summary object (from physignal)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.pls *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'pls'
summary(object, ...)
```

Arguments

object	print/summary object (from phylo.integration or two.b.pls)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

summary.procD.lm *Print/Summary Function for geomorph*

Description

Print/Summary Function for geomorph

Usage

```
## S3 method for class 'procD.lm'
summary(object, ...)
```

Arguments

object	print/summary object (from procD.lm)
...	other arguments passed to print/summary

Author(s)

Michael Collyer

two.b.pls *Two-block partial least squares analysis for Procrustes shape variables*

Description

Function performs two-block partial least squares analysis to assess the degree of association between to blocks of Procrustes shape variables (or other variables)

Usage

```
two.b.pls(A1, A2, iter = 999, seed = NULL, print.progress = TRUE)
```

Arguments

A1	A 3D array (p x k x n) containing Procrustes shape variables for the first block, or a matrix (n x variables)
A2	A 3D array (p x k x n) containing Procrustes shape variables for the second block, or a matrix (n x variables)
iter	Number of iterations for significance testing
seed	An optional argument for setting the seed for random permutations of the resampling procedure. If left NULL (the default), the exact same P-values will be found for repeated runs of the analysis (with the same number of iterations). If seed = "random", a random seed will be used, and P-values will vary. One can also specify an integer for specific seed values, which might be of interest for advanced users.
print.progress	A logical value to indicate whether a progress bar should be printed to the screen. This is helpful for long-running analyses.

Details

The function quantifies the degree of association between two blocks of shape data as defined by Procrustes shape variables using partial least squares (see Rohlf and Corti 2000). If geometric morphometric data are used, it is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA) [e.g., with gpagen]. If other variables are used, they must be input as a 2-Dimensional matrix (rows = specimens, columns = variables). It is also assumed that the separate inputs have specimens (observations) in the same order. Additionally, if names for the objects are specified, these must be the same for both datasets. The observed test value is then compared to a distribution of values obtained by randomly permuting the individuals (rows) in one partition relative to those in the other. A significant result is found when the observed PLS correlation is large relative to this distribution. In addition, a multivariate effect size describing the strength of the effect is estimated from the empirically-generated sampling distribution (see details in Adams and Collyer 2016; Adams and Collyer 2019).

The generic function, plot, produces a two-block.pls plot. This function calls plot.pls, which produces an ordination plot. An additional argument allows one to include a vector to label points. Starting with version 3.1.0, warpgrids are no longer available with plot.pls but after making

a plot, the function returns values that can be used with picknplot.shape or a combination of shape.predictor and plotRefToTarget to visualize shape changes in the plot (via warpgrids).

For more than two blocks:

If one wishes to consider 3+ arrays or matrices, there are multiple options. First, one could perform multiple two.b.pls analyses and use compare.pls to ascertain which blocks are more "integrated". Second, one could use integration.test and perform a test that averages the amount of integration (correlations) across multiple pairwise blocks. Note that performing integration.test performed on two matrices or arrays returns the same results as two.b.pls. Thus, integration.test is more flexible and thorough.

Using phylogenies and PGLS:

If one wishes to incorporate a phylogeny, phylo.integration is the function to use. This function is exactly the same as integration.test but allows PGLS estimation of PLS vectors. Because integration.test can be used on two blocks, phylo.integration likewise allows one to perform a phylogenetic two-block PLS analysis.

Notes for geomorph 3.0:

There is a slight change in two.b.pls plots with geomorph 3.0. Rather than use the shapes of specimens that matched minimum and maximum PLS scores, major-axis regression is used and the extreme fitted values are used to generate deformation grids. This ensures that shape deformations are exactly along the major axis of shape covariation. This axis is also shown as a best-fit line in the plot.

Value

Object of class "pls" that returns a list of the following:

r.pls	The correlation coefficient between scores of projected values on the first singular vectors of left (x) and right (y) blocks of landmarks (or other variables). This value can only be negative if single variables are input, as it reduces to the Pearson correlation coefficient.
P.value	The empirically calculated P-value from the resampling procedure.
Effect.Size	The multivariate effect size associated with sigma.d.ratio.
left.pls.vectors	The singular vectors of the left (x) block
right.pls.vectors	The singular vectors of the right (y) block
random.r	The correlation coefficients found in each random permutation of the resampling procedure.
XScores	Values of left (x) block projected onto singular vectors.
YScores	Values of right (y) block projected onto singular vectors.
svd	The singular value decomposition of the cross-covariances. See svd for further details.
A1	Input values for the left block.
A2	Input values for the right block.

A1.matrix	Left block (matrix) found from A1.
A2.matrix	Right block (matrix) found from A2.
permutations	The number of random permutations used in the resampling procedure.
call	The match call.

Author(s)

Dean Adams and Michael Collyer

References

Rohlf, F.J., and M. Corti. 2000. The use of partial least-squares to study covariation in shape. Systematic Biology 49: 740-753.

Adams, D.C. and M.L. Collyer. 2016. On the comparison of the strength of morphological integration across morphometric datasets. Evolution. 70:2623-2631.

Adams, D.C. and M.L. Collyer. 2019. Comparing the strength of modular signal, and evaluating alternative modular hypotheses, using covariance ratio effect sizes with morphometric data. Evolution. 73:2352-2367.

See Also

integration.test, modularity.test, phylo.integration, and compare.pls

Examples

```
data(plethShapeFood)
Y.gpa<-gpagen(plethShapeFood$land)    #GPA-alignment

#2B-PLS between head shape and food use data
PLS <-two.b.pls(Y.gpa$coords,plethShapeFood$food,iter=999)
summary(PLS)
plot(PLS)

 ### Visualize shape variation using picknplot.shape Because picknplot
 ### requires user decisions, the following example
 ### is not run (but can be with removal of #).
 ### For detailed options, see the picknplot help file
 # picknplot.shape(plot(PLS))
```

two.d.array *Convert (p x k x n) data array into 2D data matrix*

Description

Convert a three-dimensional array of landmark coordinates into a two-dimensional matrix

Usage

```
two.d.array(A, sep = ".")
```

Arguments

A	A 3D array (p x k x n) containing landmark coordinates for a set of specimens
sep	An optional argument for variable labeling, combining landmark labels (e.g., 1, 2, 3, ...) and partial dimension labels (e.g., "x", "y", and "z"), much like the paste function. The default is sep = ".", but this can be changed to any separator. One should make sure to match separators with arrayspecs if switching between matrices and arrays.

Details

This function converts a (p x k x n) array of landmark coordinates into a two-dimensional matrix (n x [p x k]). The latter format of the shape data is useful for performing subsequent statistical analyses in R (e.g., PCA, MANOVA, PLS, etc.). Row labels are preserved if included in the original array.

Value

Function returns a two-dimensional matrix of dimension (n x [p x k]), where rows represent specimens and columns represent variables.

Author(s)

Dean Adams and Emma Sherratt

See Also

arrayspecs

Examples

```
data(plethodon)
plethodon$land    #original data in the form of 3D array

two.d.array(plethodon$land)   # Convert to a 2D data matrix
```

warpRefMesh *Creates a mesh3d object warped to the mean shape*

Description

A function that uses the thin-plate spline to warp a 3D mesh into a shape defined by a set of landmark coordinates.

Usage

```
warpRefMesh(mesh, mesh.coord, ref, color = NULL, centered = FALSE)
```

Arguments

mesh	A mesh3d object (e.g. made by read.ply)
mesh.coord	A p x k matrix of 3D coordinates digitized on the ply file.
ref	A p x k matrix of 3D coordinates made by mshape
color	Color to set the ply file $material. If the ply already has color, use NULL. For ply files without color, color=NULL will be plotted as gray.
centered	Logical If the data in mesh.coords were collected from a centered mesh (see details).

Details

Function takes a 3D mesh (class mesh3d or shape3d, e.g. from read.ply) and its digitized landmark coordinates and uses the thin-plate spline method (Bookstein 1989) to warp the mesh into the shape defined by a second set of landmark coordinates, usually those of the mean shape for a set of aligned specimens. It is highly recommended that the mean shape is used as the reference for warping (see Rohlf 1998). The workflow is as follows:

1. Calculate the mean shape using mshape
2. Choose an actual specimen to use for the warping. The specimen used as the template for this warping is recommended as one most similar in shape to the average of the sample, but can be any reasonable specimen - do this by eye, or use findMeanSpec
3. Warp this specimen into the mean shape using warpRefMesh
4. Use this average mesh where it asks for a mesh= in the analysis functions and visualization functions

Users should ensure that their mesh and mesh.coord matrix are in the same scale (a common issue is that the mesh is in micrometers and coordinates are in mm or cm). Use range(mesh$vb[1:3,]) and range(mesh.coord) to check and adjust mesh.coord as necessary.

For landmark coordinates digitized with geomorph digitizing functions, centered = TRUE. This refers to the specimen being centered prior to landmark acquisition in the RGL window. For landmark data collected outside of geomorph, centered=FALSE will usually be the case. The returned mesh3d object is for use in geomorph functions where shape deformations are plotted (picknplot.shape, two.b.pls, bilat.symmetry, and plotRefToTarget).

Value

Function returns a mesh3d object, which is a list of class mesh3d (see rgl for details)

Author(s)

Emma Sherratt

References

Bookstein, F. L. 1989 Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(6):567-585.

Rohlf, F. J. 1998. On Applications of Geometric Morphometrics to Studies of Ontogeny and Phylogeny. Systematic Biology. 47:147-158.

See Also

findMeanSpec

rgl-package (used in 3D plotting)

`warpRefOutline` *Creates a 2D outline warped to the mean shape*

Description

A function to take an outline (defined by many points) and use thin-plate spline method to warp the outline into the estimated mean shape for a set of aligned specimens.

Usage

```
warpRefOutline(file, coord, ref)
```

Arguments

`file`	A .txt or .csv file of the outline point coordinates, or a .TPS file with OUTLINES= or CURVES= elements
`coord`	A p x k matrix of 2D fixed landmark coordinates
`ref`	A p x k matrix of 2D coordinates made by mshape

Details

Function takes an outline (defined by many points) with a set of fixed landmark coordinates and uses the thin-plate spline method (Bookstein 1989) to warp the outline into the shape defined by a second set of landmark coordinates, usually those of the mean shape for a set of aligned specimens. It is highly recommended that the mean shape is used as the reference for warping (see Rohlf 1998). The workflow is as follows:

1. Calculate the mean shape using mshape

2. Choose an actual specimen to use for the warping. The specimen used as the template for this warping is recommended as one most similar in shape to the average of the sample, but can be any reasonable specimen - do this by eye, or use `findMeanSpec`
3. Warp this specimen into the mean shape using `warpRefOutline`
4. Use this average outline where it asks for a outline= in the analysis functions and visualization functions

The returned outline object is for use in geomorph functions where shape deformations are plotted (`picknplot.shape`, `two.b.pls`, `bilat.symmetry`, and `plotRefToTarget`).

Value

Function returns an outline object

Author(s)

Emma Sherratt

References

Bookstein, F. L. 1989 Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(6):567-585.

Rohlf, F. J. 1998. On Applications of Geometric Morphometrics to Studies of Ontogeny and Phylogeny. Systematic Biology. 47:147-158.

See Also

`findMeanSpec`

---

writeland.tps *Write landmark data to tps file*

Description

Write *.tps file from obtain landmark coordinates in a 3-dimensional array

Usage

```
writeland.tps(A, file, scale = NULL, specID = TRUE)
```

Arguments

A	A 3D array (p x k x n) containing landmark coordinates for a set of specimens
file	Name of the *.tps file to be created
scale	An optional vector containing the length of the scale for each specimen
specID	A logical value stating whether specimen ID names should be saved to line ID=

Details

This function writes a *.tps file from a 3-dimensional array (p x k x n) of landmark coordinates.

Author(s)

Dean Adams

Index