Package ‘morphoBlocks’

August 23, 2021

Type Package

Title Constructing a Multiple-Part Morphospace Using a Multiple-Block Method

Version 0.1.0

Authors Daniel Thomas & Aaron Harmer

Maintainer Daniel Thomas <d.b.thomas@massey.ac.nz>

Description The 'morphoBlocks' package provides a workflow for constructing a multiple-part morphospace with regularised consensus principal component analysis (RCPCA) using either traditional landmarks or pseudolandmarks.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports adephylo, data.table, geomorph, graphics, grDevices, methods, Morpho, phytools, RGCCA, rgl, Rvcg, stats

RoxygenNote 7.1.1

Suggests knitr, kableExtra, rmarkdown, here

VignetteBuilder knitr

R topics documented:

analyseBlocks 2
block-class 3
blockList-class 3
combineBlocks 4
decMesh 5
dodecBlock 6
formatBlock 8
loadingsPlot 10
penguinWings 11
phylomsPlot 12
readGPSA 14
readPts 16
reMesh 17
scoresPlot 18

Index **20**

analyseBlocks — *Analyse a list of data blocks containing shape information*

Description

Performs either regularised consensus principal component analysis on a list of data blocks containing shape information, or performs principal component analysis on a superblock produced by the column-wise concatenation of the individual data blocks.

Usage

```
analyseBlocks(blockList, option = "rcpca", ncomp = 3)
```

Arguments

`blockList`	list of 'block' objects produced by the `dodecBlock`, `formatBlock`, `readPts` or `readGPSA` functions.
`option`	either `option = "rcpca"` (default) for regularised consensus principal component analysis in mode 2 applied to the entire block list, or `option = "pca"` for principal component analysis applied to the superblock item in the block list.
`ncomp`	an integer specifying how many components should be calculated (default is 3)

Details

`analyseBlocks` is applied to an object of class "blockList" produced by the `combineBlocks` function and has two options: 1) `option = "rcpca"` and 2) `option = "pca"`. The `option = "rcpca"` will perform regularised consensus principal component analysis using the `rgcca` function from the `RGCCA` package (Tenenhaus and Guillemot 2017), and is the default option for `analyseBlocks`. The `rgcca` function itself has many options that each perform a different type of analysis. Here the `analyseBlocks` function is specifically calling the regularised consensus principal component analysis in mode 2 option with scaling applied. For further detail see Tenenhaus and Guillemot (2017) and Tenenhaus et al. (2017). `option = "pca"` will perform principal component analysis on the superblock item in the block list using the `prcomp` function from `base` R.

Value

A list object containing output from the regularised consensus principal component analysis or principal component analysis. The list contains the elements:

`result`	output from the regularised consensus principal component analysis in mode 2 produced by the `rgcca` function from the `RGCCA` package, or output from principal component analysis produced by the `prcomp` function in `base` R.
`option`	either "rcpca" or "pca".
`block.list`	a list containing the data blocks and a concatenated superblock. Inherited from the supplied blockList object and retained for downstream analyses.
`scores`	component score values (for individual blocks and the consensus if `option = "rpca"`; for the superblock if `option = "pca"`) (see `scoresPlot` for more detail).
`block.loadings`	component loadings (for individual blocks and the consensus if `option = "rpca"`; for the superblock if `option = "pca"`) (see `loadingsPlot` for more detail).

`p`	number of points in the configurations of each data block. Inherited from the supplied blockList object and retained for downstream analyses.
`k`	number of dimensions that the points in each configuration has. Inherited from the supplied blockList object and retained for downstream analyses.
`n`	number of configurations included in each data block. Inherited from the supplied blockList object and retained for downstream analyses.

References

Tenenhaus A, Guillemot V. 2017. RGCCA: Regularized and sparse generalized canonical correlation analysis for multiblock data 2.1.2. https://CRAN.R-project.org/package=RGCCA.

Tenenhaus M, Tenenhaus A, Groenen PJF. 2017. Regularized generalized canonical correlation analysis: A framework for sequential multiblock component methods. Psychometrika 82: 737-777 https://doi.org/10.1007/s11336-017-9573-x

Examples

```
block1 <- dodecBlock()
block2 <- dodecBlock()
blocklist <- combineBlocks(blocks = c(block1, block2))
result1 <- analyseBlocks(blocklist)
result2 <- analyseBlocks(blocklist, option = "pca", ncomp = 10)
```

`block-class` *S4 block class*

Description

Used as input for multi-block analysis.

`blockList-class` *S4 blockList class*

Description

Used as input for multiblock analysis.

combineBlocks *Combine and scale blocks of shape configuration data*

Description

Combines two or more blocks of shape configuration data into a block list. Scales all data blocks using the normalised centroid size method from Profico et al. (2019) (and further described by Collyer et al. 2020). Such scaling occurs by default but is optional and can be prevented. A superblock is produced by column-wise concatenation of the individual blocks.

Usage

```
combineBlocks(blocks, cent.scale = TRUE)
```

Arguments

`blocks`	vector of 'block' object names produced by the `dodecBlock`, `formatBlock`, `readPts` or `readGPSA` functions. e.g. blocks = c(block1, block2, block3)
`cent.scale`	a logical value indicating if the blocks should be scaled using the normalised centroid size method adapted from `geomorph::combine.subsets`. Default value `TRUE`.

Details

The block data is scaled using the normalised centroid size method from Profico et al. (2019) and Collyer et al. (2020) if `cent.scale` = `TRUE`. Dimensions of the blocks are retained for downstream analyses. `combineBlocks` adapts the normalised centroid size method from the `combine.subsets` function in the `geomorph` package (Adams and Ot?rola-Castillo 2013).

Value

a 'blockList' object, used for downstream analyses, that contains the data blocks and a superblock formed from the column-wise concatenation of all data blocks. The list contains the elements:

`block.list`	a list containing the data blocks and a concatenated superblockblock
`p`	number of points in the configurations of each data block
`k`	number of dimensions that the points in each configuration has
`n`	number of configurations included in each data block

References

Adams DC, Ot?rola-Castillo E. 2013. geomorph: an R package for the collection and analysis of geometric morphometric shape data. Methods in Ecology and Evolution 4:393-399 https://doi.org/10.1111/2041-210X.12035

Collyer ML, Davis MA, Adams DC. 2020. Making heads or tails of combined landmark configurations in geometric morphometric data. Evolutionary Biology 47: 193-205 https://doi.org/10.1007/s11692-020-09503-z

Profico A, Piras P, Buzi C, Del Bove A, Melchionna M, Senczuk G, Varano V, Veneziano A, Raia P, Manzi G. 2019. Seeing the wood through the trees. Combining shape information from different landmark configurations. Hystrix, the Italian Journal of Mammalogy, 30: 157-165 https://doi.org/10.4404/hystrix-00206-2019

Examples

```
block1 <- dodecBlock()
block2 <- dodecBlock()
combineBlocks(blocks = c(block1, block2))
```

`decMesh` *Produce new decimated versions of a set of 3D meshes that all have the same number of evenly distributed vertices*

Description

Generates new versions of 3D meshes (.obj, .ply or .stl) in a selected directory. The vertex count in the new meshes is reduced to the smallest vertex count amongst the selected meshes. Users may scale the vertex count down further. Vertices in the new meshes are approximately uniformly distributed. Meshes are automatically exported in polygon file format (.ply). `decMesh` is useful for studies that want to reduce the vertex count of 3D meshes before applying whole-mesh analyses (e.g. Generalized Procrustes Surface Analysis from Pomidor et al. 2016).

Usage

```
decMesh(dirpath, scale = 1, vsize = 0.25)
```

Arguments

`dirpath`	the directory path where 3D meshes with .obj, .ply, or .stl extensions will be found
`scale`	a proportion for specifying how the vertex counts in the new versions of the 3D meshes should be scaled relative to the smallest vertex count amongst the 3D meshes. For scale = 1 (default), all new versions of meshes will have vertex counts that are similar to the vertex count of the smallest mesh. `decMesh` expects a scale value to be between 0 and 1
`vsize`	a constant used to start the remesh process. Default is 0.25. Should only need to be adjusted during function troubleshooting

Details

`decMesh` can be used for 3D mesh preprocessing before computationally demanding analyses. `decMesh` is a wrapper for `pcAlign` from the `Morpho` package (Schlager, 2017) and `vcgImport`, `vcgPlyWrite`, `vcgQEdecim` and `vcgUniformRemesh` from the `Rvcg` package (Schlager, 2017). Users may consider Instant Meshes as an alternative to `decMesh` (Wenzel et al. 2015).

Note

Function involves several iterative steps and can therefore be slow.

References

Pomidor BJ, Makedonska J, Slice DE. 2016. A landmark-free method for three-dimensional shape analysis. PLoS One 11: e0150368 https://doi.org/10.1371/journal.pone.0150368

Schlager S. 2017. Morpho and Rvcg-shape analysis in R. In Zheng G, Li S, Sz?kely (eds.) Statistical shape and deformation analysis. Academic Press, London. Pp. 217-256.

Wenzel J, Tarini M, Panozzo D, Sorkine-Hormung O. 2015. Instant field-alighed meshes. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2015) 34: 189-181.

Examples

```
## Not run:
# Any directory path where 3D meshes with .obj, .ply, or .stl extensions will be found
path <- "/."

# Example 1
# Vertex count in new meshes becomes the same as the smallest vertex count amongst meshes
decMesh(path)

# Example 2
# Vertex count in new meshes becomes 10 percent of smallest vertex count amongst meshes
decMesh(path, scale = 0.1)

## End(Not run)
```

---

dodecBlock *Simulate a data block of landmark configurations*

Description

Generates a block of *n* dodecahedra where the vertices of each dodecahedron represent a landmark configuration with $p = 20$ points in $k = 3$ dimensions. The data block is transformed using generalised Procrustes analysis.

Usage

```
dodecBlock(
  n = 10,
  dist_x = NULL,
  dist_y = NULL,
  dist_z = NULL,
  theta_x = NULL,
  theta_y = NULL,
  theta_z = NULL,
  size = NULL,
  noise = NULL,
  vertex_shift = NULL,
  plot = FALSE
)
```

Arguments

`n`	the number of configurations to generate for the data block. Default is 10. Must be 2 or greater.
`dist_x, dist_y, dist_z`	optional integer or vector to specify a distance along the stated axis (i.e. `dist_x` for the `x` axis) for translating the configurations. If no value is specified then configurations will not be translated. If a single integer is supplied (e.g. `dist_x = 1`) then all configurations will be translated along an axis by the same amount. If a vector of length *n* is supplied (e.g. `dist_x = 1:10`) then each configuration will be translated by a value corresponding to its position in the vector.
`theta_x, theta_y, theta_z`	optional integer or vector to specify an angle for rotating the configuration about its origin in the stated axis (i.e. `theta_x` for the `x` axis). If no value is specified then configurations will not be rotated. If a single integer is supplied (e.g. `theta_x = 30`) then all configurations will be rotated by the same amount. If a vector of length *n* is supplied (e.g. `theta_x = seq(from = 0,to = 210,by = 30)`) then each configuration will be rotated by a value corresponding to its position in the vector.
`size`	optional integer or vector for scaling the configurations. If no value is specified then configurations will not be scaled relative to one another. If a single integer is supplied (e.g. `size = 1`) then all configurations will be scaled by the same amount. If a vector of length *n* is supplied (e.g. `size = 1:10`) then each configuration will be scaled by a value corresponding to its position in the vector.
`noise`	optional integer for introducing noise to the configurations. If no value is specified then `runif(1,min = 0,max = (1 / 100))` is added to each of the `x`, `y` and `z` positions of each point in each configuration. This small amount of noise is added so that no configuration has identical coordinates. The amount of noise that is introduced to the data block can be increased by specifying a noise value that is greater than 1.
`vertex_shift`	optional integer or vector to specify a distance along the z axis for translating the first vertex in each configuration. This argument can be used to introduce variation into the data block. If no value is specified then vertex 1 in each configuration will not be translated. If a single integer is supplied (e.g. `vertex_shift = 1`) then vertex 1 in all configurations will be translated along the z axis by the same amount. If a vector of length *n* is supplied (e.g. `vertex_shift = 1:10`) then vertex 1 in each configuration will be translated by a value corresponding to its position in the vector. Things get really fun when you use both `vertex_shift` and `theta_z` together (see example 2 below).
`plot`	a logical value indicating whether the configurations should be plotted. Default is FALSE.

Details

`dodecBlock` builds a data block of configurations that have been transformed using generalised Procrustes analysis. The resulting data block is a three dimensional array with *n = 2* or greater configurations, with each configuration represented by *p = 20* points that have *k = 3* dimensions. Several objects are calculated for downstream analyses including centroid sizes for each configuration. `formatBlock` uses the `cSize` function from the `Morpho` package (Schlager 2017), and the `gpagen`, `two.d.array` and `arrayspecs` functions from the `geomorph` package (Adams and Ot?rola-Castillo 2013).

Value

a 'blockList' object of *n* dodecahedra, used for downstream analyses. The list contains the elements:

`raw`	configurations without Procrustes transformation
`gpa.3D`	configurations after Procrustes transformation organised into a 3D array
`gpa.2D`	configurations after Procrustes transformation organised into a 2D matrix
`centroid`	centroid sizes of the configurations
`p`	number of points in the configurations of each data block
`k`	number of dimensions that the points in each configuration has
`n`	number of configurations included in each data block

References

Adams DC, Ot?rola-Castillo E. 2013. geomorph: an R package for the collection and analysis of geometric morphometric shape data. Methods in Ecology and Evolution 4:393-399 https://doi.org/10.1111/2041-210X.12035

Schlager S. 2017. Morpho and Rvcg-shape analysis in R. In Zheng G, Li S, Sz?kely (eds.) Statistical shape and deformation analysis. Academic Press, London. Pp. 217-256.

Examples

```
# Example 1: Generate a block with ten configurations
block <- dodecBlock()

# Example 2: Generate a block with five configurations.
# Rotate the configurations and translate the first vertex in each configuration.
# Plot the result.
block2 <- dodecBlock(n = 5, theta_z = seq(from = 0, to = 270, by = 60), vertex_shift = 0:4, plot = TRUE)
```

`formatBlock` *Format configurations into a data block*

Description

Organises data representing a block of landmark configurations (or pseudolandmark configurations) into a single data block. `formatBlock` is useful for formatting a data block of configurations from sources other than Landmark Editor (Wiley et al. 2005) or Generalised Procrustes Surface Analysis (Pomidor et al. 2016).

Usage

```
formatBlock(
  block,
  curves = NULL,
  surfaces = NULL,
  cs = NULL,
  k = 2,
  gpa = TRUE
)
```

Arguments

`block`	two-dimensional matrix or three-dimensional array of configurations to be formatted into a data block.
`curves`	optional matrix passed to `gpagen` for correctly calculating position of curve semilandmarks (see `geomorph::gpagen` for more detail).
`surfaces`	optional vector passed to `gpagen` for defining positions of semilandmarks on surfaces (see `geomorph::gpagen` for more detail).
`cs`	optional vector of centroid sizes.
`k`	number of dimensions that each point within the data block has. Required for configurations presented as a two-dimensional matrix. `k = 2` by default, but is updated when data are presented as a three-dimensional array.
`gpa`	a logical value indicating whether generalised Procrustes analyses should be performed. Default is `TRUE`.

Details

`formatBlock` expects configurations to be presented as either a two-dimensional matrix or a three-dimensional array. If data are presented as a two-dimensional matrix then the number of *k* dimensions for the landmarks (or pseudolandmarks) must be specified (default `k = 2`), each configuration should be a separate row in the matrix, and columns should contain corresponding points and dimensions in each configuration. If data are presented as a three-dimensional array then each of the *n* configurations should be a separate matrix in the array (i.e. `array[p,k,n]`), and each matrix should have the same number of rows and columns where rows represent *p* points and columns represent *k* dimensions of each point. By default `formatBlock` will perform generalised Procustes analysis (gpa) on the block using `gpagen` from the `geomorph` package (Adams and Otárola-Castillo 2013). Curve and surface information must be supplied if they are required. Centroid size will be calculated from the original configurations before Procrustes transformation. If gpa is not required then set gpa = `FALSE`. Centroid sizes of the original configurations should be supplied using the `cs` argument if gpa = `FALSE`, or else centroid sizes will be calculated from the (presumably Procrustes-transformed) configurations that are supplied. `formatBlock` is a wrapper function for the `cSize` function from the `Morpho` package (Schlager 2017), and the `gpagen`, `two.d.array` and `arrayspecs` functions from the `geomorph` package (Adams and Otárola-Castillo 2013).

Value

a 'block' object, used for downstream analyses. The list contains the elements:

`raw`	collation of original configuration data
`gpa.3D`	if gpa = `TRUE`, configurations after Procrustes transformation organised into a 3D array. If gpa = `FALSE`, original configuration data organised into a 3D array (may duplicate `raw`)
`gpa.2D`	if gpa = `TRUE`, configurations after Procrustes transformation organised into a 2D matrix. If gpa = `FALSE`, original configuration data organised into a 2D matrix (may duplicate `raw`)
`centroid`	centroid sizes either calculated from original configurations or supplied using the `cs` argument
`p`	number of points in the configurations of each data block
`k`	number of dimensions that the points in each configuration has
`n`	number of configurations included in each data block

References

Adams DC, Otárola-Castillo E. 2013. geomorph: an R package for the collection and analysis of geometric morphometric shape data. Methods in Ecology and Evolution 4:393–399 https://doi.org/10.1111/2041-210X.12035

Schlager S. 2017. Morpho and Rvcg–shape analysis in R. In Zheng G, Li S, Székely (eds.) Statistical shape and deformation analysis. Academic Press, London. Pp. 217–256.

Pomidor BJ, Makedonska J, Slice DE. 2016. A landmark-free method for three-dimensional shape analysis. PLoS One 11: e0150368 https://doi.org/10.1371/journal.pone.0150368

Wiley DF, Amenta N, Alcantara DA, Ghosh D, Kil YJ, Delson E, Harcourt-Smith W, Rohlf FJ, St. John K, Hamann B. 2005. Evolutionary morphing. Proceedings of the IEEE Visualization 2005 (VIS'05), 431–438.

Examples

```
# Format the head and tail data from the Plethodon dataset in the geomorph library
library(geomorph)
data(larvalMorph)
block1 <- formatBlock(block = larvalMorph$headcoords, curves = larvalMorph$head.sliders)
block2 <- formatBlock(block = larvalMorph$tailcoords, curves = larvalMorph$tail.sliders)
```

`loadingsPlot` *Plot component loadings from an analysis of multiple data blocks*

Description

Two- or three-dimensional scatterplots showing the loadings from an analysis of data blocks containing shape information

Usage

```
loadingsPlot(result, comp = 1, cex.2d = 2, cex.3d = 5)
```

Arguments

`result`	result produced by the analyseBlocks function.
`comp`	the component selected to be shown in the loadings plots. Default is component 1. The selected component must be within the range of components calculated by `analyseBlocks`.
`cex.2d`	value for specifying point size when plotting loadings from analysis of two-dimensional shape configurations. Passed to cex term of `plot`. Default is 2.
`cex.3d`	value for specifying point size when plotting loadings from analysis of three-dimensional shape configurations. Passed to cex term of `plot3d`. Default is 5.

Details

`loadingsPlot` helps to visualise the result from the `analyseBlocks` function by using the component loadings to colour the mean position of each point in the consensus space (for `option = "rcpca"` in `analyseBlocks`) or the concatenated superblock (for `option = "pca"` in `analyseBlocks`). Points are coloured along a gradient from orange to blue. Points that are more orange have relatively large loadings values (i.e. larger within-block covariation), and points that are more blue have relatively small loadings values (i.e. smaller within-block covariation). `loadingsPlot` will present a two-dimensional scatterplot if the list of data blocks that were analysed had `k = 2` dimensions. Likewise, `loadingsPlot` will present a three-dimensional scatterplot if the list of data blocks had `k = 3` dimensions. Data from the individual blocks that contributed to either the consensus space or concatenated superblock are shown as separate scatterplots for visual clarity. Separate plots are produced because the points in each block have the same origin after Procrustes transformation and would plot on top of one another if they were presented along the same set of Cartesian axes. `loadingsPlot` uses the `arrayspecs` and `mshape` functions from the `geomorph` package (Adams and Otárola-Castillo 2013).

Value

a two- or three-dimensional scatterplot

References

Adams DC, Otárola-Castillo E. 2013. geomorph: an R package for the collection and analysis of geometric morphometric shape data. Methods in Ecology and Evolution 4:393–399 https://doi.org/10.1111/2041-210X.12035

Examples

```
block1 <- dodecBlock()
block2 <- dodecBlock()
blocklist <- combineBlocks(blocks = c(block1, block2))
result1 <- analyseBlocks(blocklist)
result2 <- analyseBlocks(blocklist, option = "pca", ncomp = 10)
loadingsPlot(result1)
loadingsPlot(result2, comp = 2)
```

`penguinWings` *Penguin partial wing skeleton landmarks dataset*

Description

The multi-part objects in this example are partial wing skeletons comprised of humerus, radius and ulna. These partial wing skeletons are from 15 extant species of penguin and five fossil species of penguin, which together constitute a dataset of 60 wing bones. Shape data from the multi-part objects are provided as landmark configurations. Three sets of landmarks were produced for each of the digital replicas so that the replicates could be averaged to mitigate effects of placement error. The nine sets of landmark configurations (i.e. three sets from each of the three bones) were separately read into R using `readPts` with `gpa = FALSE` (i.e. generalised Procrustes transformation not performed). Metadata for each bone are available on github (https://github.com/aharmer/morphoBlocks).

Usage

```
data(penguinWings)

hum2

hum3

rad1

rad2

rad3

uln1

uln2

uln3
```

Format

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

An object of class `block` of length 1.

`phylomsPlot` *Plot a phylomorphospace for a component from an analysis of multiple data blocks*

Description

A plot to visualise the evolutionary relationships between specimens that are represented by values projected into a morphospace (e.g. Sidlauskas, 2008). Here the projected values are from a single component from an analysis of multiple data blocks. Rather than generating a two- or three-dimensional morphospace, this function plots consensus or principal component values against tree distance to show changes in the component value across the evolutionary history of the clade. The component values that are visualised using this function will either be from 1) the consensus space of an analysis performed with regularised consensus principal component analysis, or 2) will be from an analysis of a superblock (i.e. column-wise concatenation of individual data blocks) if principal component analysis was performed instead.

Usage

```
phylomsPlot(
  result,
  phy,
  n.names,
  comp = 1,
  pcol = NULL,
  xlab = "Time (Ma)",
  label = "off",
  fsize = 0.8
)
```

Arguments

`result`	result produced by the `analyseBlocks` function.
`phy`	a phylogenetic tree (class “phylo”) with `n` tips, where `n` is the number of samples in each block. Tip labels must be included and must correspond to the names of the samples in each block.
`n.names`	a vector of character strings where entries in the vector represent names of the `n` samples present in each block. IMPORTANT NOTE: The sequence of character strings in `n.names` must match the sequence that samples are named in each block, and the names must exactly match the tip labels in the phylogenetic tree. For example, consider a block containing data from five samples where the first, second and third rows of the block contain data from samples "A", "B" and "C", and the fourth and fifth rows contain data from samples "D" and "E". Consequently, 1) `n.names = c("A","B","C","D","E")`, 2) `phy` should only include five tips, and 3) `phy$tip.labels` should be "A", "B", "C", "D" and "E" (in any order).
`comp`	the component selected to be shown in the phylomorphospace plot. Default is component one. The selected component must be within the range of components calculated by `analyseBlocks`.
`pcol`	optional colour value (integer, hex code, colour name) or vector of colour values to be applied to the points in the phylomorphospace plot. If no value is specified then points will be coloured in a gradient from black to green according to their sequence in the data blocks. If a single integer is supplied (e.g. `pcol = 1` or `pcol = "red"` or `pcol = "#ffffff"` then all points will have the same colour. If a vector of length *n* is supplied (e.g. `pcol = 1:10`) then each point will be coloured by a value corresponding to its position in the vector.
`xlab`	character string displayed along the horizontal axis of the phlyomorphospace plot. Default is "Time (Ma)".
`label`	a logical value indicating if the points in the phylomorphospace should be labelled using `n.names`. Default value is `off` (i.e. names are not shown).
`fsize`	size of the `n.names` labels if they are included in the plot. Default is 0.8.

Details

`phylomsPlot` helps to visualise the result from the `analyseBlocks` function by presenting an evolutionary context for score values from a consensus space (for `option = "rcpca"` in `analyseBlocks`) or from a concatenated superblock (for `option = "pca"` in `analyseBlocks`). `phylomsPlot` is a wrapper for `phylomorphospace` from the `phytools` package (Revell, 2012), which is in turn based on the projection of a phylogenetic tree into a morphospace by Sidlauskas (2008). The function `phylomsPlot` uses the `distRoot` function from the `adephylo` package (Jombart et al. 2010). `phylomsPlot` does not specifically use functions from the `ape` package (Paradis et al. 2004), but this package may be called to read `phy` into R.

References

Jombart T, Balloux F, Dray S. 2010. adephylo: new tools for investigating the phylogenetic signal in biological traits. Bioinformatics 26: 1907–1909 https://doi.org/10.1093/bioinformatics/btq292

Paradis E, Claude J, Strimmer K. 2004. APE: Analyses of phylogenetics and evolution in R language. Bioinformatics 20: 289–290 https://doi.org/10.1093/bioinformatics/btg412

Revell LJ. 2012. phytools: An R package for phylogenetic comparative biology (and other things). Methods in Ecology and Evolution 3: 217–223 https://doi.org/10.1111/j.2041-210X.2011.00169.x

Sidlauskas B. 2008. Continuous and arrested morphological diversification in sister clades of characiform fishes: a phylomorphospace approach. Evolution 62: 3135–3156 https://doi.org/10.1111/j.1558-5646.2008.00519.x

Examples

```
# Simulate a phylogenetic tree with 20 tips
library(phytools)
phy <- pbtree(b = 0.1, n = 20)

# Simulate, combine and analyse two data blocks with 20 samples each
block1 <- dodecBlock(n = 20)
block2 <- dodecBlock(n = 20)
blocklist <- combineBlocks(blocks = c(block1, block2))
result <- analyseBlocks(blocklist)

# Simulate a vector of names (here just use the tip labels)
n.names <- phy$tip.label

# Generate phylomorphospace plot
phylomsPlot(result, phy, n.names, comp = 1, xlab = "Tip to root distance")
```

`readGPSA` *Read pseudolandmarks into a data block*

Description

Reads and organises pseudolandmarks from Generalised Procrustes Surface Analysis (GPSA; Pomidor et al. 2015) into a data block. `readGPSA` expects a directory containing a single .dat file containing homologised points for one block of pseudolandmark configurations.

Usage

```
readGPSA(dirpath)
```

Arguments

`dirpath`	the directory path where a file containing homologised pseudolandmark points will be found. The homologised points file will have a .dat extension and must be produced using the Generalised Procrustes Surface Analysis software tool (Pomidor et al. 2015).

Details

`readGPSA` reads a matrix of pseudolandmark configurations from a .dat file into the R environment and organises the configurations into an array. Several objects are calculated for downstream analyses including centroid sizes for each pseudolandmark configuration. Note that centroid sizes are not calculated for the original meshes processed using GPSA, but are instead calculated for the homologised pseudolandmark points from each of those meshes after processing with GPSA. `readGPSA` is a wrapper function for the `fread` function from the `data.table` package (Dowle and Srinivasan 2020), the `cSize` function from the `Morpho` package (Schlager 2017), and the `arrayspecs` function from the `geomorph` package (Adams and Otárola-Castillo 2013).

Value

a 'block' class object, used for downstream analyses. The list contains the elements:

`raw`	collation of pseudolandmark configurations organised as a 2D matrix
`gpa.3D`	pseudolandmark configurations organised into a 3D array
`gpa.2D`	duplication of `raw`. The duplication occurs so that the object structure is consistent with the structure produced by closely-related functions (e.g. `readPts`)
`centroid`	centroid sizes of pseudolandmark configurations
`p`	number of pseudolandmarks that each configuration within the data block has
`k`	number of dimensions that each pseudolandmark within the data block has
`n`	number of pseudolandmark configurations included in the data block

References

Adams DC, Otárola-Castillo E. 2013. geomorph: an R package for the collection and analysis of geometric morphometric shape data. Methods in Ecology and Evolution 4:393–399 https://doi.org/10.1111/2041-210X.12035

Dowle M, Srinivasan A. 2020. data.table: Extension of 'data.frame'. R package version 1.13.4. https://CRAN.R-project.org/package=data.table

Pomidor BJ, Makedonska J, Slice DE. 2016. A landmark-free method for three-dimensional shape analysis. PLoS One 11: e0150368 https://doi.org/10.1371/journal.pone.0150368

Schlager S. 2017. Morpho and Rvcg–shape analysis in R. In Zheng G, Li S, Székely (eds.) Statistical shape and deformation analysis. Academic Press, London. Pp. 217–256.

Examples

```
## Not run:
# For this example to work a directory (/...) containing .dat file must first be prepared.
dirpath <- "/..."
block1 <- readGPSA(dirpath)
block1@p
block1@k
block1@n
```

```
## End(Not run)
```

readPts	*Read and combine multiple Landmark Editor files into a single data block*

Description

Reads, collates and transforms landmark configurations from multiple specimens. `readPts` expects a directory containing two or more files with .pts extensions (i.e. landmark configurations exported from the Landmark Editor, Wiley et al. 2005).

Usage

```
readPts(dirpath, landmarkRM = c(), gpa = TRUE)
```

Arguments

`dirpath`	the directory path where two or more landmark configurations with .pts extensions will be found.
`landmarkRM`	a vector of landmarks to be excluded from the data block.
`gpa`	a logical value indicating whether generalised Procrustes analyses should be performed. Default is `TRUE`.

Details

`readPts` reads landmark configurations from .pts files into the R environment, organises the configurations into a single array, and performs generalised Procrustes analysis on the array if required. Several objects are calculated for downstream analyses including centroid sizes for each landmark configuration. `readPts` is a wrapper function for the `read.pts` and `cSize` functions from the `Morpho` package (Schlager 2017), and the `gpagen`, `two.d.array` and `arrayspecs` functions from the `geomorph` package (Adams and Otárola-Castillo 2013). Landmarks are identified by their sequence within the configuration, which is an important consideration when excluding landmarks from the data block. For example, `landmarkRM = c(1,3)` would remove the first and third landmarks from all configurations once they were read into the R environment, and thus the data block would not include landmark 1 and landmark 3. The `landmarkRM` term might be used for analyses that want to test the sensitivity of dataset covariation on one or more landmarks.

Value

a 'block' class object, used for downstream analyses. The list contains the elements:

`raw`	collation of landmark configurations without Procrustes transformation
`gpa.3D`	landmark configurations after Procrustes transformation organised into a 3D array
`gpa.2D`	landmark configurations after Procrustes transformation organised into a 2D matrix
`centroid`	centroid sizes of landmark configurations without Procrustes transformation
`p`	number of landmarks that each configuration within the data block has

`k`	number of dimensions that each landmark within the data block has
`n`	number of landmark configurations included in the data block
`curves`	matrix for correctly calculating position of curve semilandmarks (see `geomorph::gpagen` for more detail)
`surfaces`	vector passed for defining positions of semilandmarks on surfaces (see `geomorph::gpagen` for more detail)

References

Adams DC, Otárola-Castillo E. 2013. geomorph: an R package for the collection and analysis of geometric morphometric shape data. Methods in Ecology and Evolution 4:393–399 https://doi.org/10.1111/2041-210X.12035

Schlager S. 2017. Morpho and Rvcg–shape analysis in R. In Zheng G, Li S, Székely (eds.) Statistical shape and deformation analysis. Academic Press, London. Pp. 217–256.

Wiley DF, Amenta N, Alcantara DA, Ghosh D, Kil YJ, Delson E, Harcourt-Smith W, Rohlf FJ, St. John K, Hamann B. 2005. Evolutionary morphing. Proceedings of the IEEE Visualization 2005 (VIS'05), 431–438.

Examples

```
## Not run:
# Example 1
# For this example to work a directory (/...) containing .pts files must first be prepared.
dirpath <- "/..."
block1 <- readPts(dirpath)
block1@p
block1@k
block1@n

# Example 2
# Exclude the first and third landmarks from the data block
block2 <- readPts(dirpath, landmarkRM = c(1, 3))
block2@p
block2@k
block2@n

## End(Not run)
```

`reMesh` *Produce new versions of 3D meshes with similar numbers of evenly distributed vertices*

Description

Generates new versions of 3D meshes (.obj, .ply or .stl) in a selected directory. The vertex count in the new meshes is reduced to become similar to the smallest vertex count amongst the selected meshes. Vertices in the new meshes are uniformly distributed. Meshes are automatically exported in polygon file format (.ply). `reMesh` is useful for studies that want to apply whole-mesh analyses (e.g. Generalized Procrustes Surface Analysis from Pomidor et al. 2016) to a set of 3D meshes that have very large vertex counts, or have large disparity in vertex counts between meshes.

Usage

```
reMesh(dirpath, scale = 1)
```

Arguments

`dirpath`	the directory path where two or 3D meshes with .obj, .ply, or .stl extensions will be found.
`scale`	a proportion for specifying how the vertex counts in the new versions of the 3D meshes should be scaled relative to the smallest vertex count amongst the 3D meshes. For `scale = 1` (default), all new versions of meshes will have vertex counts that are similar to the vertex count of the smallest mesh.

Details

`reMesh` can be used for 3D mesh preprocessing before computationally demanding analyses. `reMesh` is a wrapper for `vcgImport`, `vcgMeshres`, `vcgPlyWrite` and `vcgQEdecim` from the `Rvcg` package (Schlager, 2017).

References

Pomidor BJ, Makedonska J, Slice DE. 2016. A landmark-free method for three-dimensional shape analysis. PLoS One 11: e0150368 https://doi.org/10.1371/journal.pone.0150368

Schlager S. 2017. Morpho and Rvcg–shape analysis in R. In Zheng G, Li S, Székely (eds.) Statistical shape and deformation analysis. Academic Press, London. Pp. 217–256.

`scoresPlot` *Plot component scores from an analysis of multiple data blocks*

Description

Bivariate scatterplot (or plots) of the score values from an analysis of data blocks containing shape information

Usage

```
scoresPlot(
  result,
  comp = c(1, 2),
  pcol = NULL,
  plabels = NULL,
  consensus.only = FALSE
)
```

Arguments

`result`	result produced by the analyseBlocks function.
`comp`	the components selected to be shown along the scatter plot axes. Default is component one and component two. The first selected component will be shown along the horizontal axis and the second selected component will be shown along the vertical axis. The selected components must be within the range of components calculated by `analyseBlocks`.

`pcol`	optional colour value (integer, hex code, colour name) or vector of colour values to be applied to the points in the scatterplot. If no value is specified then points will be coloured in a gradient from black to green according to their sequence in the data blocks. If a single integer is supplied (e.g. `pcol = 1` or `pcol = "red"` or `pcol = "#ffffff"`) then all points will have the same colour. If a vector of length `n` is supplied (e.g. `pcol = 1:10`) then each point will be coloured by a value corresponding to its position in the vector.
`plabels`	optional user-supplied vector of labels for labelling points in scores plot.
`consensus.only`	a logical value (default value `FALSE`), relevant only if analyses were performed using regularised consensus principal component analysis (i.e. `analyseBlocks`, `option = 'rcpca'`). If `TRUE`, only plot the global scores and the consensus space.

Details

`scoresPlot` helps to visualise the result from the `analyseBlocks` function and gives a different result depending on whether 1) `option = "rcpca"` or 2) `option = "pca"`) was used for `analyseBlocks`. 1) If `option = "rpcca"` was used for `analyseBlocks` then a scatterplot will be produced for the selected components from each individual block and for the consensus space. Axes will display the average variance explained by the components of the individual blocks and the consensus. Average variance explained by a selected component is presented as a proportion of the total variance. For detail about average variance explained see Tenenhaus and Guillemot (2017) and Tenenhaus et al. (2017). 2) If `option = "pca"` was selected for `analyseBlocks` then a scatterplot will be produced for the selected components from the analysis of the superblock. Axes will display the variance explained by the components of the superblock. Variance explained by a selected component is presented as a proportion of the total variance explained by all components.

Value

a two-dimensional scatterplot

References

Tenenhaus A, Guillemot V. 2017. RGCCA: Regularized and sparse generalized canonical correlation analysis for multiblock data 2.1.2. https://CRAN.R-project.org/package=RGCCA.

Tenenhaus M, Tenenhaus A, Groenen PJF. 2017. Regularized generalized canonical correlation analysis: A framework for sequential multiblock component methods. Psychometrika 82: 737-777 https://doi.org/10.1007/s11336-017-9573-x

Examples

```
block1 <- dodecBlock()
block2 <- dodecBlock()
blocklist <- combineBlocks(blocks = c(block1, block2))
result1 <- analyseBlocks(blocklist)
result2 <- analyseBlocks(blocklist, option = "pca", ncomp = 10)
scoresPlot(result1)
dev.off()
scoresPlot(result2,
  comp = c(2, 3),
  pcol = colorRampPalette(c(rgb(1, 0.7, 0, 1), rgb(0, 0, 1, 1)),
  alpha = TRUE)(result2$n[1]))
```

Index

∗ **datasets**
    penguinWings, 11

analyseBlocks, 2

block-class, 3
block_out (block-class), 3
blockList-class, 3
blockList_out (blockList-class), 3

combineBlocks, 4

decMesh, 5
dodecBlock, 6

formatBlock, 8

hum1 (penguinWings), 11
hum2 (penguinWings), 11
hum3 (penguinWings), 11

loadingsPlot, 10

penguinWings, 11
phylomsPlot, 12

rad1 (penguinWings), 11
rad2 (penguinWings), 11
rad3 (penguinWings), 11
readGPSA, 14
readPts, 16
reMesh, 17

scoresPlot, 18

uln1 (penguinWings), 11
uln2 (penguinWings), 11
uln3 (penguinWings), 11