Quick Guide to Geomorph v.2.0

Emma Sherratt (emma.sherratt@gmail.com)
[document version 1: 14 April 2014]
# Table of Contents

## Introduction .......................................................... 4
- What is geomorph? .................................................... 4
- How to use this manual .............................................. 4
- Workflows for common analyses .................................... 6
- Example datasets in geomorph ..................................... 7
- Permutation tests ...................................................... 8

## Section 1: Data Input: Importing landmark data ................. 9
  1.1 TPS files (readland.tps) ........................................ 9
  1.2 NTS files (readland.nts) ....................................... 9
  1.3 Multiple NTS files of single specimens (readmulti.nts) .... 10
  1.4 Morphologika files (read.morphologika) .................... 10
  1.5 Other text files .................................................. 12

## Section 2: Data Preparation: Manipulating landmark data and classifiers 14
  2.1 Data object formats ............................................. 14
    - 3D array \((p \times k \times n)\) .................................. 14
    - 2D array \((n \times [p \times k])\) ................................. 14
  2.2 Converting a 2D array into a 3D array (arrayspecs) ........ 14
  2.3 Converting a 3D array into a 2D array (two.d.array) ....... 15
  2.4 Making a factor: group variables ............................. 15
  2.5 Estimating missing landmarks (estimate.missing) .......... 16
  2.6 Rotate a subset of 2D landmarks to common articulation angle (fixed.angle) ........................................ 18

## Section 3a: Generalized Procrustes Analysis ..................... 19
  3.1 Generalized Procrustes Analysis (gpagen) .................... 19
  3.2 Generalized Procrustes Analysis with Bilateral Symmetry Analysis (bilat.symmetry) ................................. 23

## Section 3b: Data Analysis ........................................... 26
  3.3 Covariation methods: Procrustes ANOVA/regression for shape data (procD.lm) ................................. 26
  3.4 Covariation methods: Pairwise Group Comparisons (pairwiseD.test) .................................................. 27
  3.5 Covariation methods: Two-block partial least squares analysis for shape data (two.b.pls) ...................... 28
  3.6 Morphological Integration methods: Quantify morphological integration between two modules (morphol.integr) .................................................. 29
  3.7 Morphological Integration methods: Compare modular signal to alternative landmark subsets (compare.modular.partitions) .................................................. 32
  3.8 Phylogenetic Comparative methods: Assessing phylogenetic signal in morphometric data (physignal) 33
  3.9 Phylogenetic Comparative methods: Quantify phylogenetic morphological integration between two sets of variables (phylol.pls) ........................................ 36
  3.10 Phylogenetic Comparative methods: Comparing rates of shape evolution on phylogenies (compare.evols) .................................................. 37
  3.11 Calculate morphological disparity for one or more groups (morphol.disparity) .................................... 38
  3.12 Quantify and compare shape change trajectories (trajectory.analysis) ............................................... 39

## Section 4: Visualization .............................................. 42
  4.1 Principal Components Analysis (plotTangentSpace) ........ 42
  4.2 Plot allometric patterns in landmark data (plotAllometry) .................................................. 44
  4.3 Plot phylogenetic tree and specimens in tangent space (plotGMPhyloMorphoSpace) ................................. 47
4.4 Create a mesh3d object warped to the mean shape (warpRefMesh) ................................................................. 47
findMeanSpec .................................................................................................................................................. 49
4.5 Plot landmark coordinates for all specimens (plotAllSpecimens) .............................................................. 49
4.6 Plot shape differences between a reference and target specimen (plotRefToTarget) ............................... 50
4.7 Plot 3D specimen, fixed landmarks and surface semilandmarks (plotspec) ........................................... 55

Section 5: Data Collection (Digitizing) ................................................................................................................. 56
5.1 2D data collection (digitize2d) ...................................................................................................................... 56
5.2 Define sliding semilandmarks in 2D (define.sliders.2d) ............................................................................. 57
5.3 Importing 3D surface files (read.ply) ............................................................................................................ 59
5.4 3D data collection: landmarks (digit.fixed) ............................................................................................... 60
5.5 3D data collection: landmarks and semilandmarks .................................................................................. 62
buildtemplate ...................................................................................................................................................... 62
editTemplate ....................................................................................................................................................... 65
digitsurface ......................................................................................................................................................... 66
5.6 Define sliding semilandmarks in 3D (define.sliders.3d) ........................................................................... 67

Frequently Asked Questions ............................................................................................................................... 69

References ................................................................................................................................................................. 70
**Introduction**

**What is **geomorph**?**

*Geomorph* is a freely available software package for geometric morphometric analyses of two- and three-dimensional landmark (shape) data in the *R* statistical computing environment. It can be installed from the Comprehensive *R* Archive Network, CRAN [http://cran.r-project.org/web/packages/geomorph/](http://cran.r-project.org/web/packages/geomorph/).

How to cite: When using *geomorph* in publications, please cite the software with version and the publication. Type in *R* console

```
> citation(package="geomorph")
```


**How to use this manual**

This manual is not meant to be exhaustive – the benefit of working within the *R* environment is its flexibility and infinite possibilities. Instead, the manual presents the functions in *geomorph* and how they can be used together to perform analyses to address a variety of questions in Biology, Anthropology, Paleontology, Archaeology, Medicine etc. This help guide is structured according to the pipeline outlined in Figure 1, which is based on a general workflow for morphometric analysis.

![Figure 1](image-url)  
*Figure 1 Overview of the morphometric analysis process. In blue are the steps performed in *R* and *geomorph*, and those in orange are done outside of *R* and imported in.*

In section 1, we go over how to import data files of (raw) landmark coordinates digitized elsewhere, e.g., using software such as ImageJ or tpsDig for 2D data, or IDAV Landmark editor, AMIRA, Microscribe for 3D data (note that data collection – digitizing landmarks – can also be done in *geomorph*, and is outlined in section 5).
In section 2, we demonstrate some techniques and functions for preparing and manipulating imported datasets, such as adding grouping variables and estimating missing data, and adjusting articulated datasets (2D only). Note that some functions described in this section can also be used on Procrustes coordinate data, but are presented here because they are important steps to learn familiarize the user with the R environment.

In section 3a, the raw data are taken through the morphometric-specific step of alignment using a generalized Procrustes superimposition, which is imperative for raw coordinate data. In section 3b, the statistical analysis functions are presented in order by type of analysis (Table 1). In section 4, we describe how to plot and visualize the data analysis results, including shape deformation graphs and ordination plots (e.g., PCA).

Table 1 Functions in geomorph

<table>
<thead>
<tr>
<th>Data Input</th>
<th>Data Manipulation &amp; Preparation</th>
<th>Data Analysis</th>
<th>Visualization</th>
<th>Dataset</th>
<th>Digitizing</th>
</tr>
</thead>
<tbody>
<tr>
<td>read.morphologika</td>
<td>arrayspecs</td>
<td>gpagen</td>
<td>plotAllometry</td>
<td>hummingbirds</td>
<td>buildtemplate</td>
</tr>
<tr>
<td>readland.nts</td>
<td>define.modules</td>
<td>bilat.symmetry</td>
<td>plotAllSpecimens</td>
<td>mosquito</td>
<td>digitize2d</td>
</tr>
<tr>
<td>readland.tps</td>
<td>estimate.missing</td>
<td>procD.lm</td>
<td>plotGMorphoSspace</td>
<td>motionpaths</td>
<td></td>
</tr>
<tr>
<td>readmulti.nts</td>
<td>findMeanSpec</td>
<td>pairwiseD.test</td>
<td>plotRefToTarget</td>
<td>plethodon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mshape</td>
<td>two.b.pls</td>
<td>plotspec</td>
<td>digitsurface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>two.d.array</td>
<td>compare.modular.partitions</td>
<td>plotTangentSpace</td>
<td>plethspecies</td>
<td>editTemplate</td>
</tr>
<tr>
<td></td>
<td>writeLand.tps</td>
<td>morphol.integ</td>
<td>warpRefMesh</td>
<td>read.ply</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fixed.angle</td>
<td>phylo.pls</td>
<td></td>
<td>scallopPLY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>physignal</td>
<td></td>
<td>scallops</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>compare.eval.rates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>morphol.disparity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>trajectory.analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Section 5 details the functions that can be used to generate coordinate data from 2D images and 3D surface files (i.e., an ASCII .ply). And finally, at the end there are some frequently asked questions and their solutions.

Throughout this manual, we will use the following abbreviations as is conventional in morphometrics and R:

- **n**: number of specimens/individuals
- **p**: number of landmarks
- **k**: number of dimensions
- **#**: a comment, in R this is text that is ignored (not run)
- **...**: data not shown
- **>**: denotes the line of code to be written in (the ‘>’ is not typed)
- **dim()**: a function
code to be written into the R console

[1] in a code example at the start of a line, a number in brackets denotes the first element of the output and is not intended to be typed

Briefly understanding functions; below is a `geomorph` function annotated by color:

```r
plotAllometry(A, sz, groups = NULL, method = c("CAC", "RegScore", "PredLine"), warpgrids = TRUE, iter = 99, label = FALSE, mesh = NULL, verbose = FALSE)
```

**the function name** | an object, usually data, sometimes a file name | a multipart option, requires choice of one of the presented values | a logical option that requires a TRUE or FALSE input | option that is NULL by default, but requires either an object or a value | option that requires a value

Usually only the objects are necessary to run a function, as it will use the defaults for the options (which are presented in the function as above, and under “usage” in the R help pages). Always read the help pages and check the examples for usage. Order does not matter as long as the option is written in full, e.g., `A= mydata`. But “ ” are important, e.g., `method = "RegScore"`. 
Workflows for common analyses

Below are some pathways to perform common analyses in geomorph. This is not an exhaustive list, but provides a reference for users familiar with other morphometric software to navigate the functions. In red the type of question or analysis is presented, and in blue the specific *geomorph* functions in sequence.
Example datasets in *geomorph*

For sections 2 through 4, many of the examples will be using data included with *geomorph*. There are nine datasets: *plethodon*, *scallops*, *hummingbirds*, *mosquito*, *ratland*, *plethspecies*, *plethShapeFood*, *motionpaths* and *scallopPLY*. To load type,

```
> data(plethodon)
```

It is advised to run and examine these example datasets before performing own analyses in order to understand how a function and its options work, and how one’s datasets should be formatted.

> plethodon

```
$land
[1,] [,1] [,2]
[1,]  8.89372  53.77644
[2,]  9.26840  52.77072
[3,]  5.56104  54.21028
[4,]  1.87340  52.75100
...

$links
[1,] [,2]
[1,]  4  5
[2,]  3  5
[3,]  2  4
[4,]  1  2
[5,]  1  3
[6,]  6  7
...

$species
[35] Teyah Teyah Teyah Teyah Teyah Teyah
Levels: Jord Teyah

$site
Levels: Allo Symp
```

The dataset above, *plethodon*, is a list containing four components: the coordinate data (*plethodon$land*), the wirelink addresses for plotting (*plethodon$links*), and two sets of grouping variables as factors (*plethodon$species*, *plethodon$site*).

> class(plethodon$site)
```
[1] "factor"
```

> class(plethodon$land)
```
[1] "array"
```

> class(plethodon$links)
```
[1] "matrix"
```

Many of the novice user problems when using *geomorph* and *R* stem from having the object input in the wrong format. Here are some useful base functions in *R* to help understand formatting of one’s data:

```
class() # Object Classes
attributes() # Object Attribute Lists
dim() # Dimensions of an Object
nrow() ; ncol() # The Number of Rows/Columns of a 2D array
dimnames() # Dimnames of an Object
names() # The Names of an Object
rownames(); colnames() # Row and Column Names
```
is.numeric() # very useful to know if the data are numeric or not – dataframes often do not preserved the data as numeric even though they are numbers

**Geomorph** primarily has data stored in a 2D array or 3D array (*matrix* and *array* respectively) (see section 2.1), grouping variables are *vectors* and *factors*, and outputs of functions may be *lists*. For more information about the object classes in R see [http://www.statmethods.net/input/datatypes.html](http://www.statmethods.net/input/datatypes.html)

**Permutation tests**
Many of the function in **geomorph** test for statistical significance using a permutation procedure (e.g., Good 2000). A randomization test takes the original data, shuffles and resamples, calculates the test statistic and compares this to the original. This is repeated for a number of iterations, creating a distribution of random tests statistics in which the original can be evaluated. The proportion of random samples that provide a better fit to the data than the original provides the *P*-value. Therefore the number of decimal places for the *P*-value is correlated to the number of iterations. When deciding how many iterations to use more is better. However there is a point where it is time consuming and not helpful (Adams and Anthony 1996). The default in **geomorph** is 999, up to 10,000 is reasonable. The examples in this manual and on the package help files are usually very low to make them fast to run, and it is not recommended to run the function at these small iterations for one’s own data.

Finally, this manual only covers **geomorph** functions. It is recommended that users look to some “getting started with R” resources, such as Quick-R ([http://www.statmethods.net/](http://www.statmethods.net/)), and the R Introduction manual on ([http://www.r-project.org/](http://www.r-project.org/)), and various Springer eBooks in the series ‘Use R!’ . Also highly recommended is J. Claude’s book *Morphometric with R* (2008).
Section 1: Data Input: Importing landmark data

1.1 TPS files (readland.tps)

**Function:**

```
readland.tps(file, specID = c("None", "ID", "imageID"))
```

This function reads a *.tps file containing two- or three-dimensional landmark coordinates for a set of specimens. 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 (there is no line “SCALE=”), landmarks are treated in their original units. The name of the specimen can be given in the tps file by “ID=” (use specID="ID") or “IMAGE=” (use specID= “imageID”), otherwise the function defaults to specID= “None”.

```
ratland.tps
LM=8
-0.45 -0.475
-0.59 -0.28
-0.515 -0.12
-0.33  0
 0  0
0.145 -0.395
-0.045 -0.42
-0.26 -0.465
ID = specimen103N
LM=8
...

> mydata <- readland.tps("ratland.tps", specID = "ID")
[1] "Not all specimens have scale. Using scale = 1.0"
> mydata[,1]
[,1] [,2]
[1,] -0.450 -0.475
[2,] -0.590 -0.280
[3,] -0.515 -0.120
[4,] -0.330  0.000
[5,]  0.000  0.000
[6,]  0.145 -0.395
[7,] -0.045 -0.420
[8,] -0.260 -0.465
```

In this case, there is no scale given in the tps file, so the command warns that the data are treated in their original units. The function returns a 3D array containing the coordinate data, and if provided in the file, the names of the specimens (dimnames(mydata)[[3]])

1.2 NTS files (readland.nts)

**Function:**

```
readland.nts(file)
```

Function reads *.nts file containing a matrix of two- or three-dimensional landmark coordinates for a set of specimens. NTS files are text files in one of the standard formats for geometric morphometrics (see Rohlf 2012). 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. 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. Here there are n = 44 and p*k = 50 (25 2D landmarks).

### rats.nts

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.450</td>
<td>-0.475</td>
<td>-0.590</td>
<td>-0.280</td>
<td>-0.515</td>
<td>-0.120</td>
<td>-0.330</td>
<td>0</td>
<td>0</td>
<td>0.145</td>
</tr>
<tr>
<td>-0.330</td>
<td>-0.395</td>
<td>-0.405</td>
<td>-0.420</td>
<td>-0.260</td>
<td>-0.465</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.530</td>
<td>-0.555</td>
<td>-0.685</td>
<td>-0.320</td>
<td>-0.625</td>
<td>-0.120</td>
<td>-0.400</td>
<td>0</td>
<td>0</td>
<td>0.230</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.560</td>
<td>-0.670</td>
<td>-0.700</td>
<td>-0.590</td>
<td>-0.280</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.500</td>
<td>-0.745</td>
<td>-0.355</td>
<td>-0.700</td>
<td>-0.435</td>
<td>-0.120</td>
<td>-0.400</td>
<td>0</td>
<td>0</td>
<td>0.300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.650</td>
<td>-0.800</td>
<td>-0.435</td>
<td>-0.715</td>
<td>-0.900</td>
<td>-0.450</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.360</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The function returns a 3D array containing the coordinate data, and if provided in the file, the names of the specimens (dimnames(mydata)[[3]]).

Function is for *.nts file containing landmark coordinates for multiple specimens. Note that *.dta files in the nts format written by Landmark Editor [http://graphics.idav.ucdavis.edu/research/projects/EvoMorph](http://graphics.idav.ucdavis.edu/research/projects/EvoMorph), and *.nts files written by Stratovan Checkpoint [http://www.stratovan.com/](http://www.stratovan.com/) 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. NAs will be introduced unless the header is manually.

### 1.3 Multiple NTS files of single specimens (readmulti.nts)

**Function:**

```r
readmulti.nts(file)
```

This function reads a list containing the names of multiple *.nts files, where each contains the landmark coordinates for a single specimen. For these files, the number of variables (columns) of the data matrix will equal the number of dimensions of the landmark data (k=2 or 3). When the function is called a dialog box is opened, from which the user may select multiple *.nts files. These are then read and concatenated into a single matrix for all specimens.

```r
> filelist <- list.files(pattern = ".nts")
> mydata <- readmulti.nts(filelist)
```

The function returns a 3D array containing the coordinate data, and the names of the specimens (dimnames(mydata)[[3]]) extracted from the file names.

### 1.4 Morphologika files (read.morphologika)

**Function:**

```r
read,morphologika(file)
```
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), see http://sites.google.com/site/hymsfme/resources. If the headers "[labels]" and "[labelvalues]" are present in the file, 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.

```
morphologikaexample.txt
[[individuals]
15
[landmarks]
31
[Dimensions]
3
[names]
Specimen 1
Specimen 2
Specimen 3
...
Specimen 15
[labels]
Sex
[labelvalues]
Female
Female
...
Female
[rawpoints]
' #1
16.01 24.17 11.18
15 24.86 11.16
14.96 25.54 11.52
16.26 24.36 11.48
15.89 26.61 11.83
17.16 25.33 12.35
18.22 23.65 11.12
...
```

> mydata <- read.morphologika("morphologikaexample.txt")

> mydata$coords[,,1]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16.01 24.17 11.18</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15 24.86 11.16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14.96 25.54 11.52</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16.26 24.36 11.48</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15.89 26.61 11.83</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>17.16 25.33 12.35</td>
<td></td>
</tr>
</tbody>
</table>

> mydata$labels

```
Specimen 1 "Female"
Specimen 2 "Female"
Specimen 3 "Female"
Specimen 4 "Female"
Specimen 5 "Male"
Specimen 6 "Male"
...
```

The function returns a 3D array containing the coordinate data, and if provided, the names of the specimens (dimnames(mydata)[[3]])). If other optional headers are present in the file (e.g., "[labels]" or "[wireframe]"), function returns a list containing the 3D array of coordinates ($coords), and a data matrix of the data from...
labels ($labels$) and/or the landmark addresses demoting the wireframe ($wireframe$) – which can be passed to see plotRefToTarget() option 'links').

1.5 Other text files

Base Functions:

```r
read.table(file)
read.csv(file)
```

Using base read functions in $R$, one can read in data by many other ways. Here are two examples. These examples use data arrangement function arrayspecs() (see section 2.1 for details) and creates an object in the same way as the previous functions.

For a set of files (file1.txt, file2.txt, file3.txt...) each containing the landmark coordinates of a single specimen

```r
file1.txt
16.01 24.17 11.18
15 24.86 11.16
14.96 25.54 11.52
16.26 24.36 11.48
15.89 26.61 11.83
17.16 25.33 12.35
18.22 23.65 11.12
...
```

```r
> filelist <- list.files(pattern = "*.txt") # makes a list of all .txt files in working directory
> names <- gsub (".txt", "", filelist) # extracts names of specimens from the file name
> coords = NULL # make an empty object
> for (i in 1:length(filelist)){
  tmp <- as.matrix(read.table(filelist[i]))
  coords <- rbind(coords, tmp)
}
> coords <- arrayspecs(coords, p, k)
> dimnames(coords[[3]]) <- names
```

For a files each containing the landmark coordinates of a set of specimens, where each row is a specimen, and coordinate data arranged in columns x1, y1, x2, y2... etc., and the first column is the ID of the specimens, e.g., from a data file exported from MorphoJ.

```r
coordinatedata.txt
ID X1 Y1 X2 Y2 X3 Y3 ...
specimen1 0.595 0.1679 0.2232 0.5028 1.292 0.4237 0.51 ...
specimen2 0.0038 1.3925 0.7966 0.4132 0.1006 0.8483 ...
specimen3 0.6249 0.4515 0.3576 1.3262 0.9114 0.3611 ...
...
```

```r
> tmp <- as.matrix(read.table("coordinatedata.txt", header=T))
> names <- tmp[,1]
> coords <- arrayspecs(tmp[,2:ncol(tmp)], p, k)
> dimnames(coords)[[3]] <- names
```

If it is a data file exported from MorphoJ, and the centroid size and/or classifiers are included as columns before the data, follow the example above and replace the “2” on coords<- line with the column number the coordinate data start on. Then one can make a second matrix of the classifier data by:
> classifiers <- tmp[,1:x] # taking only the columns containing classifiers, where x is the number of the last column before the coordinate data.
Section 2: Data Preparation: Manipulating landmark data and classifiers

2.1 Data object formats
Landmark data in geomorph can be found as objects in two formats: a 2D array (matrix; Figure 2A) or a 3D array (Figure 2B). These data formats follow the convention in other morphometric packages (e.g., shapes, Morpho) and in J.Claude’s book Morphometrics in R (2008).

3D array (p x k x n)
An array with three dimensions, i.e., number of rows (p), number of columns (k) and number of “sheets” (n). Imagine a 3D array like a stack of filing cards. Data in this format are needed for most geomorph analysis functions. If one has inputted data using

```
readland.nts(), readmulti.nts(), readland.tps(), read.morphologika()
```
Then the data will be a 3D array object. Check by typing

```
> dim(mydata)
[1] 15 3 40
```
If `dim()` gives three numbers, it is a 3D array. Here `mydata` has p=15, k=3, n=40.

If `dim()` gives two number, it is a 2D array (a matrix).

2D array (n x [p x k])
An array (matrix) with two dimensions, i.e., number of rows (n) and number of columns (p*k). From the example above

```
> dim(mydata)
[1] 40 30
```
Data in this format are needed for the following geomorph analysis functions (see section 3):

```
procD.lm(), trajectory.analysis(), pairwise.D.test()
```

2.2 Converting a 2D array into a 3D array (arrayspecs)
Function:

```
arrayspecs(A, p, k)
```
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.

```r
> A <- arrayspecs(mydata, p, k)  # where mydata is a 2D array
> A[,1] # look at just the first specimen
  [,1]     [,2]  
[1,]  8.893720 53.77644 
[2,]  9.267994 52.77972 
[3,]  8.036040 54.21028 
[4,]  1.873400 52.75100 
[5,]  1.281800 53.18484 
[6,]  1.242360 53.32288 
[7,]  0.847960 54.70328 
[8,]  3.352400 55.76816 
[9,]  6.290680 55.79000 
[10,]  8.874000 55.25544 
[11,] 10.747400 55.43292 
[12,] 14.395600 52.75100
```

### 2.3 Converting a 3D array into a 2D array (two.d.array)

**Function:**

```
two.d.array(A)
```

This function converts a 3D array (p x k x n) of landmark coordinates into a 2D array (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.

```r
> a <- two.d.array(mydata)  # where mydata is a 3D array
> a
[1,]  8.893720 53.77644  9.267994 52.77972  8.036040 54.21028  1.873400 52.75100  1.281800 53.18484
[2,]  8.679000 54.57819  8.935628 53.83027  5.451914 54.65691  1.987882 52.68871  1.515514 53.02331
[4,]  9.637164 58.03294  9.952104 56.77318  6.109836 57.94896  2.645496 55.89135  2.015616 56.62621
...
```

### 2.4 Making a factor: group variables

Many analyses will require a grouping variable (a classifier) for the data. For small datasets, this can be made easily within R:

```r
> group <- factor(c(0,0,1,0,0,1,1,0,0))  # specimens assigned in order to group 0 or 1
> names(group) <- dimnames(mydata)[[3]]  # assign specimen names from 3D array of data to the group classifier
> group
[1] 0 0 1 0 0 1 1 0 0
Levels: 0 1
```
If the data have many specimens or many different groups, it may be easier to make a table in excel, save as a .csv file and import using read.csv().

```
> classifier <- read.csv("classifier.csv", header=T, row.names=1)
> is.factor(classifier$Habitat)  # check that it is a factor
[1] TRUE
> classifier$Habitat
[1] dry wet dry dry wet wet wet wet dry ...
Levels: dry wet
```

### 2.5 Estimating missing landmarks (estimate.missing)

All analysis and plotting functions in geomorph require a full complement of landmark coordinates. Either the missing values are estimated, or subsequent analyses are performed on a subset dataset excluding specimens with missing values. Below is the function to estimate missing data, followed by steps of how to just exclude specimens with missing values.

**Function:**

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

**Arguments:**

- `A` A 3D array (p x k x n) containing landmark coordinates for a set of specimens
- `method` Method for estimating missing landmark locations

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).
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).

Missing landmarks in a target specimen are designated by NA in place of the x,y,z coordinates. To make this so,

```r
> any(is.na(mydata)) # check if there are NAs in the data
FALSE # if false then,
> mydata[which(mydata == -999)] <- NA #change missing values from “-999” to NAs
```

Here is an example using Plethodon dataset,

```r
> data(plethodon)
> plethland<-plethodon$land
> plethland[2,,2]<-plethland[6,,2]<-NA #create missing landmarks
> plethland[2,,10]<-NA
> estimate.missing(plethland,method="TPS")
> estimate.missing(plethland,method="Reg")
```

The function returns a 3D array with the missing landmarks estimated.

Instead of estimating missing, an alternative is to proceed with the specimens for which data are missing excluded. For example to make a dataset of only the complete specimens (starting with the dataset as 2D array), two ways are possible:

```r
> mydata
[1,]  8.893720 53.77644  9.268400 52.77072
[2,]  8.679762 54.57819  8.935628 53.83027
[3,]  9.805328 56.06903    NA       NA
[4,]  9.637164 58.93294  9.952104 56.77318
[5,] NA       NA  11.335110 57.85184
[6,]  7.946625 55.71114  8.476400 54.82112
[7,]  8.849841 58.66961  9.396387 57.82877
[8,]  9.331504 56.36904 10.154872 55.31344
> newdata <- mydata[complete.cases(mydata),] # keep only specimens with complete data
> # OR
> newdata <- na.omit(mydata) # use only specimens without NAs
> newdata
[1,]  8.893720 53.77644  9.268400 52.77072
[2,]  8.679762 54.57819  8.935628 53.83027
[3,]  9.637164 58.93294  9.952104 56.77318
[4,]  7.946625 55.71114  8.476400 54.82112
[5,]  8.849841 58.66961  9.396387 57.82877
[6,]  9.331504 56.36904 10.154872 55.31344
```

These same functions can be used to make a dataset of only the landmarks in all specimens, by inputting the matrix mydata in transpose, e.g., t(mydata). Note that these methods will re-label the specimen or landmark numbers.
2.6 Rotate a subset of 2D landmarks to common articulation angle (fixed.angle)

A function for rotating a subset of landmarks so that the articulation angle between subsets is constant. Presently, the function is only implemented for two-dimensional landmark data.

*Function:*

\[
\text{fixed.angle}(A, \text{art.pt} = \text{NULL}, \text{angle.pts} = \text{NULL}, \text{rot.pts} = \text{NULL}, \\
\text{angle} = 0, \text{degrees} = \text{FALSE})
\]

*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**: A vector containing numbers specifying which two points used to define the angle (one per subset).
- **rot.pts**: A vector containing numbers specifying which landmarks are in the subset to be rotated.
- **angle**: An optional value specifying the additional amount by which the rotation should be augmented (in radians).
- **degrees**: A logical value specifying whether the additional rotation angle is expressed in degrees or radians (radians is default).

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.

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

```r
> data(plethspecies)
> fixed.angle(plethspecies$land, art.pt=1, angle.pts=c(5,6), rot.pts=c(2,3,4,5))
```

![Before and After diagrams](before_after.png)

Function returns a 3D array containing the newly rotated data.
Section 3a: Generalized Procrustes Analysis

3.1 Generalized Procrustes Analysis (gpagen)

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; Mitteroecker and Gunz 2009; Zelditch et al. 2012; 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 (Dryden and Mardia 1993; Rohlf 1999), which are used for subsequent multivariate analyses. Additionally, any semilandmarks on curves and 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).

The first step in any geometric morphometric analysis is to perform a superimposition of the raw coordinate data (read in using functions in section 1).

Function:

\[
gpagen(A, \text{Proj} = \text{TRUE}, \text{ProcD} = \text{TRUE}, \text{ShowPlot} = \text{TRUE}, \text{curves} = \text{NULL}, \text{surfaces} = \text{NULL}, \text{pointscale} = 1)
\]

Arguments:

- **A**: A 3D array (p x k x n) containing landmark coordinates for a set of specimens.
- **Proj**: A logical value indicating whether or not the aligned Procrustes residuals should be projected into tangent space.
- **ProcD**: A logical value indicating whether or not Procrustes distance should be used as the criterion for optimizing the positions of semilandmarks.
- **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 (see `define.sliders.2d()` or `define.sliders.3d()`).
- **pointscale**: An optional value defining the size of the points for all specimens.
- **surfaces**: An optional vector defining which landmarks should be treated as semilandmarks on surfaces.
- **ShowPlot**: A logical value indicating whether or not a plot of Procrustes residuals should be displayed.

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. To include semilandmarks on curves, one must specify a matrix defining which landmarks are to be treated as semilandmarks using the "curves=" option (this matrix can be made using `define.sliders.2d()` or `define.sliders.3d()`, see section 5). 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=TRUE" option will slide the semilandmarks along their tangent directions using the Procrustes distance criterion, while "ProcD=FALSE" will slide the semilandmarks based on minimizing bending energy. The aligned Procrustes residuals can be projected into tangent space using the "Proj=TRUE" option. NOTE: Large datasets may exceed the memory limitations of R.

If the curve semilandmarks were defined by `define.sliders.2d()` or `define.sliders.3d()`, or the surface semilandmarks digitized with `buildtemplate()` and `digitizesurface()` the .csv files these functions make must be first read in as follows:

```r
> curves <- as.matrix(read.csv("curveslide.csv", header=T))
> sliders <- as.matrix(read.csv("surfslide.csv", header=T))
```
Example using fixed points only:

```r
> data(plethodon)
> Y <- gpagen(plethodon$land)
> Y

$coords
[,1] [,2]
[1,]  0.18705239 -0.023704219
[2,]  0.21178322 -0.089958949
[3,] -0.03236886  0.004834644
[4,] -0.27528209 -0.091255419
[5,] -0.31422144 -0.062671167
[6,] -0.31685794 -0.053529677
[7,] -0.34273719  0.037340225
...

$Csize
...
```

The function returns a list containing the Procrustes coordinates ($coords$), the centroid sizes ($Csize$) and, if `ShowPlot=T` a graph of the specimens (grey) around the mean (black) shape:

![Procrustes Shape Plot](image)

Example using fixed points and semilandmarks on curves:

```r
> data(hummingbirds)
> hummingbirds$curvepts # Matrix defining which points are semilandmarks (middle column) and in which directions they slide (columns 1 vs. 3)

before slide after
[1,]  1  11  12
[2,]  11  12  13
[3,]  13  14  15
[4,]   7  15  14
[5,]  12  13  14
[6,]   1  16  17
[7,]  16  17  18
[8,]  17  18  19
...
```

```r
#Using Procrustes Distance for sliding
> Y <- gpagen(hummingbirds$land,curves=hummingbirds$curvepts)
> Y

$coords
```
Example using fixed points, curves and surfaces:

```r
> data(scallops)
> scallops$curvslide # Matrix defining which points are semilandmarks (middle column) and in which directions they slide (columns 1 vs. 3)
```

```
  before slide after
[1,]   5   6   7
[2,]   6   7   8
```
For 3D data, the Procrustes data are plotted in the rgl window.

For all subsequent analyses, the Procrustes coordinates ($Y$ coords) should be used.
3.2 Generalized Procrustes Analysis with Bilateral Symmetry Analysis (bilat.symmetry)

If the data has bilateral symmetry, the first step is to perform a superimposition of the raw coordinate data taking into account the symmetry. This function also assesses the statistical differences in the symmetric data.

Function:

```r
bilat.symmetry(A, ind = NULL, side = NULL, replicate = NULL, object.sym = FALSE, land.pairs = NULL, warpgrids = TRUE, mesh = NULL, verbose = FALSE)
```

Arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A 3D array (p x k x n) containing GPA-aligned coordinates for a set of specimens [for &quot;object.sym=FALSE, A is of dimension (n x k x 2n)]</td>
</tr>
<tr>
<td>ind</td>
<td>A vector containing labels for each individual. For matching symmetry, the matched pairs receive the same label (replicates also receive the same label).</td>
</tr>
<tr>
<td>side</td>
<td>An optional vector (for matching symmetry) designating which object belongs to which 'side-group'</td>
</tr>
<tr>
<td>replicate</td>
<td>An optional vector designating which objects belong to which group of replicates</td>
</tr>
<tr>
<td>object.sym</td>
<td>A logical value specifying whether the analysis should proceed based on object symmetry =TRUE or matching symmetry =FALSE</td>
</tr>
<tr>
<td>land.pairs</td>
<td>An optional matrix (for object symmetry) containing numbers for matched pairs of landmarks across the line of symmetry</td>
</tr>
<tr>
<td>warpgrids</td>
<td>A logical value indicating whether deformation grids for directional and fluctuating components of asymmetry</td>
</tr>
<tr>
<td>mesh</td>
<td>A mesh3d object to be warped to represent shape deformation of the directional and fluctuating components of asymmetry if warpgrids= TRUE (see warpRefMesh).</td>
</tr>
<tr>
<td>verbose</td>
<td>A logical value indicating whether the output is basic or verbose</td>
</tr>
</tbody>
</table>

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 and Goodall's F tests (i.e., an isotropic model of shape variation). 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). Analytical and computational details concerning the analysis of symmetry in geometric morphometrics can be found in Mardia et al. (2000) and Klingenberg et al. (2002).

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.
Example of matching symmetry:

```r
data(mosquito)
bilat.symmetry(mosquito$wingshape, ind=mosquito$ind, side=mosquito$side, replicate=mosquito$replicate, object.sym=FALSE, verbose=F)
```

$\text{ANOVA.size}$

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ind</td>
<td>9</td>
<td>4.149667e-09</td>
<td>4.610741e-10</td>
<td>0.5964824</td>
</tr>
<tr>
<td>side</td>
<td>1</td>
<td>3.473794e-10</td>
<td>3.473794e-10</td>
<td>0.4493978</td>
</tr>
<tr>
<td>ind:side</td>
<td>9</td>
<td>6.956898e-09</td>
<td>7.729887e-10</td>
<td>1.4170492</td>
</tr>
<tr>
<td>replicate</td>
<td>20</td>
<td>1.090984e-08</td>
<td>5.454918e-10</td>
<td>NA</td>
</tr>
</tbody>
</table>

$\text{ANOVA.Shape}$

<table>
<thead>
<tr>
<th>df</th>
<th>SS.obs</th>
<th>MS</th>
<th>F,Goodall</th>
<th>P,param</th>
<th>F, P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ind</td>
<td>288</td>
<td>0.104888121</td>
<td>0.0003641949</td>
<td>0.5</td>
<td>0.45532641</td>
</tr>
<tr>
<td>side</td>
<td>32</td>
<td>0.003220857</td>
<td>0.0001006518</td>
<td>1.0</td>
<td>0.01398196</td>
</tr>
<tr>
<td>ind:side</td>
<td>288</td>
<td>0.038990419</td>
<td>0.0001353834</td>
<td>1.0</td>
<td>0.16926004</td>
</tr>
<tr>
<td>ind:side:replicate</td>
<td>640</td>
<td>0.083258692</td>
<td>0.0001300917</td>
<td>1.0</td>
<td>0.36143160</td>
</tr>
</tbody>
</table>

Symmetric Shape Component (left) and Asymmetric Shape Component (right)

Function returns the ANOVA table for analysis of symmetry and a graph showing the shape deformations relating to the symmetric and asymmetric components of shape. When verbose=TRUE, the function returns the symmetric component of shape variation ($\$\text{symm.shape}$) and the asymmetric component of shape variation ($\$\text{asymm.shape}$), to be used in subsequent analyses like Procrustes coordinates.
Example of object symmetry:

```r
> data(scallops)
> bilat.symmetry(scallops$coorddata, ind=.scallops$ind, object.sym=TRUE,
land.pairs=scallops$land.pairs)
   df SS.obs           MS F.Goodall    P.param         F  P
ind 276 0.064029966 2.319926e-04       0.5 0.64430682  8.837028  0
side 62 0.028837521 4.651213e-04       0.5 0.29017994 17.717330  0
ind:side 248 0.006510579 2.625234e-05     1.0 0.06551324  NA  NA
```

![Directional Asymmetry](image1)
![Fluctuating Asymmetry](image2)
Section 3b: Data Analysis

After the data have been superimposed with `gpagen()` or `bilat.symmetry()`, the Procrustes coordinates can be used in many statistical analyses (this section), ordination methods and visualization methods (Section 4).

![Diagram showing Procrustes coordinates and covariation methods](image)

**Protip!** Throughout the analysis functions, one will see the option `verbose=TRUE/FALSE`. This is an option to limit the amount of information returned. `FALSE` is default, and the basic results are returned, usually the test statistic and \( P \)-value. When `TRUE`, the function will also return new datasets, and shape deformation coordinates that can be used in other functions. Look out for this option in many of the functions.

Several functions have the option “mesh=”, which is there for 3D data users that wish to view the shape deformations as a warped surface mesh. Further details in `warpRefMesh()`.

3.3 Covariation methods: Procrustes ANOVA/regression for shape data (procD.lm)

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

**Function:**

```
procD.lm(f1, data = NULL, iter = 999)
```

**Arguments:**

- **f1** A formula for the linear model (e.g., \( y \sim x1+x2 \)), where \( y \) is a **two-dimensional array** of shape data
- **data** An optional value specifying a data frame containing all data (not required)
The function quantifies the relative amount of shape variation attributable to one or more factors in a linear model and assesses this variation via permutation. Data input is specified by a formula (e.g., \( y \sim X \)), where 'y' specifies the response variables (shape data), and 'X' contains one or more independent variables (discrete or continuous). The response matrix 'y' must be in the form of a 2D array rather than a 3D array. 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.

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, where the rows of the shape matrix are randomized relative to the design matrix. Procedurally, Procrustes ANOVA is identical to permutational-MANOVA as used in other fields (Anderson 2001). For several reasons, Procrustes ANOVA is particularly useful for shape data. First, covariance matrices from GPA-aligned Procrustes coordinates are singular, and thus standard approaches such as MANOVA cannot be accomplished unless generalized inverses are utilized. This problem is accentuated when using sliding semilandmarks. Additionally, geometric morphometric datasets often have more variables than specimens (the 'small N large P' problem). In these cases, distance-based procedures can still be utilized to assess statistical hypotheses, whereas standard linear models cannot.

**Example of a allometric regression, \( y \sim x \) (continuous):**

```r
> data(ratland)
> rat.gpa<-gpagen(ratland)        #GPA-alignment
> procD.lm(two.d.array(rat.gpa$coords)~rat.gpa$Csize,iter=99)
   df  SS.obs       MS       F  P.val   Rsq
rat.gpa$Csize   1 0.6403531 0.640353114 513.4641  0.0 1.0
Total        163 0.8423871 0.005168019       NA   NA   NA
```

This example tests for the association between centroid size and shape. If it is significant, we suggest examining the allometric shape variation using `plotAllometry()`.

**Example of a MANOVA for Goodall's F test, \( y \sim x \) (discrete):**

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
> y<-two.d.array(Y.gpa$coords)
> procD.lm(y~plethodon$species*plethodon$site,iter=99)
   df  SS.obs       MS       F   P.val   Rsq
plethodon$species                 1 0.02925784 0.029257838 14.54367  0.0 0.1485624
plethodon$site                    1 0.06437484 0.064374844 31.99985  0.0 0.3268759
plethodon$species:plethodon$site  1 0.03088502 0.030885020 15.35252  0.0 0.1568247
Total                39 0.19693973 0.005049737       NA   NA   NA
```

Visualization of the shape changes can be done with `plotRefToTarget()`.

### 3.4 Covariation methods: Pairwise Group Comparisons (pairwiseD.test)
The function is designed as a post-hoc test to Procrustes ANOVA (3.3), where the latter has identified significant shape variation explained by a grouping factor.

**Function:**

```r
pairwiseD.test(y, x, iter = 999)
```

**Arguments:**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A two-dimensional array of shape data</td>
<td>A factor defining groups</td>
<td>Number of iterations for permutation test</td>
</tr>
</tbody>
</table>

The function performs pairwise comparisons to identify shape among groups. The function takes as input the shape data (y), and a grouping factor (x). It then estimates the Euclidean distances among group means, which are used as test values. These are then statistically evaluated through permutation, where the rows of the shape matrix are randomized relative to the grouping variable. The input for the shape data (y) must be in the form of a 2D array rather than a 3D array.

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
> y<-two.d.array(Y.gpa$coords)
> procD.lm(y~plethodon$species,iter=99) # Procrustes ANOVA
> pairwiseD.test(y,plethodon$species,iter=99)
```

The function returns a list with a matrix of the Euclidean distances among groups ($Dist.obs$) and a matrix of the pairwise significance levels based on permutation ($Prob.dist$).

### 3.5 Covariation methods: Two-block partial least squares analysis for shape data (two.b.pls)

Function performs two-block partial least squares analysis to assess the degree of association between to blocks of Procrustes-aligned coordinates (or other variables).

**Function:**

```r
two.b.pls(A1, A2, warpgrids = TRUE, iter = 999, verbose = FALSE)
```

**Arguments:**

<table>
<thead>
<tr>
<th>x</th>
<th>A1</th>
<th>A2</th>
<th>iter</th>
<th>warpgrids</th>
<th>verbose</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 2D array (n x [p1 x k]) or 3D array (p1 x k x n) containing GPA-aligned coordinates for Block1</td>
<td>A 2D array (n x [p2 x k]) or 3D array (p2 x k x n) containing GPA-aligned coordinates for Block2</td>
<td>Number of iterations for significance testing</td>
<td>A logical value indicating whether deformation grids for shapes along PC1 should be displayed (only relevant if data for A1 or A2 [or both] were input as 3D array)</td>
<td>A logical value indicating whether the output is basic or verbose (see Value below)</td>
<td></td>
</tr>
</tbody>
</table>

The function quantifies the degree of association between two blocks of shape data as defined by landmark coordinates (or one block of shape data and a block of continuous, multivariate data) using partial least squares (see Rohlf and Corti 2000).
An example of 2B-PLS of head shape against diet,

```r
> data(plethShapeFood)
> plethShapeFood$food
oligoch gastrop isopoda diplop chilop acar araneida chelon coleopt
[1,] -0.333567 -0.2116501 -0.3740649 -0.2462653 -0.2774757 0.2035222 -0.3740685 -0.3089659 0.71842496
[2,] -0.333567 3.0823626 -0.2116501 3.5516887 -0.5007004 -0.3740685 -0.3089659 -1.39084544
[3,] -0.333567 -0.2116501 -0.3740649 -0.2462653 -0.2774757 0.2035222 2.1697963 -0.3089659 0.71842496
[4,] -0.333567 -0.2116501 -0.3740649 -0.2462653 -0.2774757 0.2035222 2.1697963 -0.3089659 0.71842496

...```

A plot of PLS scores from Block1 versus Block2 is provided for the first set of PLS axes. Thin-plate spline deformation grids along these axes are also shown (if data were input as a 3D array and warpgrids=TRUE). If verbose = TRUE, the function also returns a list containing the PLS scores for the first block of landmarks ($Xscores$) and for the second block of landmarks ($Yscores$). These can be used to make further plots (plot()) and shape deformations (plotRefToTarget()).

**Protip!** Function can be used in many combinations, and is not exclusively for shape data. For example, one block as shape data for a structure and the other block shape for another structure (but also see morphol.integr()), or one block as multivariate continuous data (e.g., Precipitation) and other block for shape data or two blocks of non-shape, multivariate continuous data.

### 3.6 **Morphological Integration methods: Quantify morphological integration between two modules (morphol.integr)**

Function quantifies the degree of morphological integration between two modules of Procrustes-aligned coordinates. The function may be used to assess the degree of morphological integration between two separate structures or between two modules defined within the same landmark configuration.

**Function:**

```r
morphol.integr(A1, A2, method = c("PLS", "RV"), warpgrids = TRUE, iter = 999, verbose = FALSE)
```

**Arguments:**

- **A1** A 2D array (n x [p1 x k]) or 3D array (p1 x k x n) containing GPA-aligned coordinates for the first module
- **A2** A 2D array (n x [p2 x k]) or 3D array (p2 x k x n) containing GPA-aligned coordinates for the second module
Two analytical approaches are currently implemented to assess the degree of morphological integration, method="PLS" and method="RV".

**method="PLS"** (default) the function estimates the degree of morphological integration using two-block partial least squares, or PLS. When used with landmark data, this analysis is referred to as singular warps analysis (Bookstein et al. 2003). When method="PLS", the scores along the X & Y PLS axes are also returned, as is a plot of PLS scores from Block1 versus Block2 along the first set of PLS axes. Thin-plate spline deformation grids along these axes are also shown (if data were input as a 3D array). Note: deformation grids are displayed for each block of landmarks separately.

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
> morphol.integr(Y.gpa$coords[1:5,,],Y.gpa$coords[6:12,,],method="PLS",iter=99)
[1] "No specimen names in data matrix 1. Assuming specimens in same order."
$PLS.corr
[1] 0.9108678

$pvalue
[1] 0.01
```

A plot of PLS scores from Block1 versus Block2 is provided for the first set of PLS axes. Thin-plate spline deformation grids along these axes are also shown (if data were input as a 3D array and warpgrids=TRUE). If verbose=TRUE, the function also returns a list containing the PLS scores for the first block of landmarks ($Xscores$) and for the second block of landmarks ($Yscores$). These can be used to make further plots (plot()) and shape deformations (plotRefToTarget()).

**Protip!** If the two blocks of landmarks are derived from a single structure (i.e., a single landmark configuration), one can plot the overall deformation along PLS1 as follows,
> res<-morphol.integr(Y.gpa$coords[1:5,,],Y.gpa$coords[6:12,,],
    method="PLS",iter=99, verbose=TRUE)
> ref<-mshape(Y.gpa$coords)   #overall reference
> plotRefToTarget(ref,Y.gpa$coords[,which.min(res$x.scores)],method="TPS") #Min
  along PLS1
> plotRefToTarget(ref,Y.gpa$coords[,which.max(res$x.scores)],method="TPS") #Max
  along PLS1

**method="RV"** the function estimates the degree of morphological integration using the $RV$ coefficient
(Klingenberg 2009). Significance testing for both approaches is found by permuting the objects in one data
matrix relative to those in the other. A histogram of coefficients obtained via resampling is presented, with
the observed value designated by an arrow in the plot.

> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
> morphol.integr(Y.gpa$coords[1:5,,],Y.gpa$coords[6:12,,],method="RV",iter=99)
[1] "No specimen names in data matrix 1. Assuming specimens in same order."
$RV
[1] 0.6214977

$pvalue
[1] 0.01

If evaluating an a priori hypothesis of modularity within a structure is of interest, one may use the average
$RV$ coefficient as implemented in the function `compare.modular.partitions()` 3.7.
3.7  Morphological Integration methods: Compare modular signal to alternative landmark subsets 
(compare.modular.partitions)

Function quantifies the degree of morphological integration between two or more modules of Procrustes-aligned landmark coordinates defined \textit{a priori} and compares this to patterns found by randomly assigning landmarks into subsets. Only modules within the same configuration should be tested.

\textit{Function:}

\begin{verbatim}
compare.modular.partitions(A, landgroups, iter = 999)
\end{verbatim}

\textit{Arguments:}

- \texttt{A}  A 3D array (p x k x n) containing GPA-aligned coordinates for all specimens
- \texttt{landgroups} A list of which landmarks belong in which partition (e.g., A,A,A,B,B,B,C,C,C)
- \texttt{iter} Number of iterations for significance testing

The function quantifies the degree of morphological integration between two or more modules of shape data as defined by landmark coordinates, and compares this to modular signals found by randomly assigning landmarks to modules. The degree of morphological integration is quantified using the \textit{RV} coefficient (Klingenberg 2009). If more than two modules are defined, the average \textit{RV} coefficient is utilized (see Klingenberg 2009). The \textit{RV} 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 \textit{RV} coefficient is small relative to this distribution (see Klingenberg 2009). A histogram of coefficients obtained via resampling is presented, with the observed value designated by an arrow in the plot.

The \texttt{landgroups} argument can be made by hand, using \texttt{c()}, or there is a graphical assisted function \texttt{define.modules()}.  

\textit{Function:}

\begin{verbatim}
define.modules(spec, nmodules)
\end{verbatim}

\textit{Arguments:}

- \texttt{spec} Name of specimen, as an object matrix containing 2D landmark coordinates
- \texttt{nmodules} Number of modules to be defined

Function takes a matrix of two-dimensional digitized landmark coordinates and allows user assign landmarks to each module. The output is a list of which landmarks belong in which partition, to be used by \texttt{compare.modular.partitions()}. The number of modules is chosen by the user (up to five).

\texttt{define.modules}

Example where the modularity hypothesis is the cranium versus the mandible,

\begin{verbatim}
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)    #GPA-alignment
> #landmarks on the skull and mandible assigned to partitions
> land.gps <- c("A","A","A","A","A","B","B","B","B","B","B","B","B")
> # OR
> land.gps <- define.modules(plethodon$land[,,1], nmodules=2)
\end{verbatim}
Select landmarks in module 1
Press esc when finished

Select landmarks in module 2
Press esc when finished

[1] "A" "A" "A" "A" "A" "B" "B" "B" "B" "B" "B" "B"
> compare.modular.partitions(Y.gpa$coords, land.gps, iter=99)
> #Result implies that the skull and mandible are not independent modules

3.8 Phylogenetic Comparative methods: Assessing phylogenetic signal in morphometric data (physignal)

Function calculates the degree of phylogenetic signal from a set of Procrustes-aligned specimens.

Function:
physignal(phy, A, iter = 249, c("Kmult", "SSC"))

Arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phy</td>
<td>A phylogenetic tree of class 'phylo'</td>
</tr>
<tr>
<td>A</td>
<td>A 2D array (n x [p x k]) or 3D array (p x k x n) containing GPA-aligned coordinates for a set of specimens</td>
</tr>
<tr>
<td>iter</td>
<td>Number of iterations for significance testing</td>
</tr>
<tr>
<td>method</td>
<td>XXX</td>
</tr>
</tbody>
</table>

The function estimates the degree of phylogenetic signal present in shape data for a given phylogeny. Two approaches may be used to quantify phylogenetic signal. First, method = "Kmult", a multivariate version of the K-statistic may be utilized (Kmult: Adams 2014a). 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 2014a).

The second approach, method = "SSC", estimates the phylogenetic signal as the sum of squared changes (SSC) in shape along all branches of the phylogeny (Klingenberg and Gidaszewski 2010). Significance testing is found by permuting the shape data among the tips of the phylogeny. Note that the method can be slow as ancestral states must be estimated for every iteration.

A plot of the specimens in tangent space with the phylogeny superimposed is included.

The tree must have number of tips equal to number of taxa in the data matrix (e.g., ape package function drop.tip()). And, tip labels of the tree MUST be exactly the same as the taxa names in the landmark data matrix (check using match()). To learn more about phylogenetic trees in R, look at these resources:

http://www.r-phylo.org/wiki/ and
http://bodegaphylo.wikispot.org/Phylogenetics_and_Comparative_Methods_in_R

> data(plethspecies)
> Y.gpa<-gpagen(plethspecies$land) #GPA-alignment
> plethspecies$phy # look at the structure of the tree (class phylo)

Phylogenetic tree with 9 tips and 8 internal nodes.

Tip labels:

P_serratus, P_cinereus, P_shenandoah, P_hoffmani, P_virginia, P_nettingi, ...

Rooted; includes branch lengths.
> plot(plethspecies$phy)
Example using method = "Kmult",
> physignal(plethspecies$phy,Y.gpa$coords,iter=99, method = "Kmult")
$phy.signal
[,1]
[1,] 0.957254
$pvalue
[,1]
[1,] 0.02

Example using method = “SSC” ,
> physignal(plethspecies$phy,Y.gpa$coords,iter=99, method = "SSC")
$phy.signal
[1] 0.002235759
$pvalue
[1] 0.01

This function also calls plotGMPhyloMorphoSpace() and plots the phylomorphospace.

This function can be used with univariate data (i.e., centroid size) if imported as matrix with row names giving the taxa names.

> # make matrix Csize with names
> Csize <- matrix(Y.gpa$Csize, dimnames=list(names(Y.gpa$Csize)))
> physignal(plethspecies$phy,Csize,iter=99, method = "Kmult")
$phy.signal
[,1]
[1,] 0.7097642
$pvalue
[,1]
[1,] 0.49

> physignal(plethspecies$phy,Csize,iter=99, method = "SSC")
$phy.signal
[1] 1.762033e-07
$pvalue
[1] 0.3
3.9 Phylogenetic Comparative methods: Quantify phylogenetic morphological integration between two sets of variables (phylo.pls)

Function quantifies the degree of phylogenetic morphological covariation between two sets of Procrustes-aligned coordinates using partial least squares.

**Function:**

```r
phylo.pls(A1, A2, phy, warpgrids = TRUE, iter = 999, verbose = FALSE)
```

**Arguments:**

- **A1** A 2D array (n x [p1 x k]) or 3D array (p1 x k x n) containing landmark coordinates for the first block
- **A2** A 2D array (n x [p2 x k]) or 3D array (p2 x k x n) containing landmark coordinates for the second block
- **phy** Object of class ‘phylo’ with tip labels corresponding to the row names of the blocks of data
- **warpgrids** A logical value indicating whether deformation grids for shapes along PC1 should be displayed (only relevant if data for A1 or A2 [or both] were input as 3D array)
- **iter** Number of iterations for significance testing
- **verbose** A logical value indicating whether the output is basic or verbose (see Value below)

The function estimates the degree of morphological covariation between two sets of variables while accounting for phylogeny using partial least squares (Adams and Felice 2014). The observed value is statistically assessed using permutation, where data for one block are permuted across the tips of the phylogeny, an estimate of the covariation between sets of variables, and compared to the observed value. A plot of PLS scores from Block1 versus Block2 is provided for the first set of PLS axes. Thin-plate spline deformation grids along these axes are also shown (if data were input as a 3D array).

```r
> data(plethspecies)
> Y.gpa<-gpagen(plethspecies$land) #GPA-alignment
> phylo.pls(Y.gpa$coords[1:5,,],Y.gpa$coords[6:11,,],plethspecies$phy,iter=5)

$`PLS Correlation`
[1] 0.9338463

$pvalue

[,1]
[1,] 0.02777778

A plot of PLS scores from Block1 versus Block2 is provided for the first set of PLS axes. Thin-plate spline deformation grids along these axes are also shown (if data were input as a 3D array and warpgrids=TRUE). If verbose=TRUE, the function also returns a list containing the PLS scores for the first block of landmarks ($Xscores) and for the second block of landmarks ($Yscores). These can be used to make further plots (plot()) and shape deformations (plotRefToTarget()).
3.10 Phylogenetic Comparative methods: Comparing rates of shape evolution on phylogenies (compare.evol.rates)

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

Function:

\[
\text{compare.evol.rates(phy, A, gp, iter = 999)}
\]

Arguments:

- **phy**: A phylogenetic tree of class 'phylo'
- **A**: A matrix (n x [p x k]) or 3D array (p x k x n) containing GPA-aligned coordinates for a set of specimens
- **gp**: A factor array designating group membership
- **iter**: Number of iterations for significance testing

The function compares rates of morphological evolution for two or more groups of species on a phylogeny, under a Brownian motion model of evolution. The approach is based on the distances between species in morphospace after phylogenetic transformation (Adams 2014b). From the data the rate of shape evolution for each group 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 2014b). Significance testing is accomplished by phylogenetic simulation in which tips data are obtained under Brownian motion using a single evolutionary rate for all species on the phylogeny. If three or more groups of species are used, pairwise \( P \)-values are also returned. A histogram of evolutionary rate ratios obtained via phylogenetic simulation is presented, with the observed value designated by an arrow in the plot. The function can be used to obtain a rate for the whole dataset of species by using a dummy group factor assigning all species to one group.

Example, comparing endangered versus not endangered species of Plethodon,

\[
\text{data(plethspecies)}
\]
\[
\text{Y.gpa<-gpagen(plethspecies$land) \quad \#GPA-alignment}
\]
\[
\text{gp.end<-factor(c(0,0,1,0,0,1,1,0,0)) \quad \#endangered species vs. rest}
\]
\[
\text{names(gp.end)<-plethspecies$phy$tip}
\]
\[
\text{compare.evol.rates(plethspecies$phy,Y.gpa$coords,gp=gp.end,iter=49)}
\]
\[
\text{$\sigma$}
\]
\[
\text{d}
\]
\[
\begin{bmatrix}
1.297943e-06
\end{bmatrix}
\]
\[
\text{$\sigma$}
\]
\[
\text{mad}
\]
\[
\begin{bmatrix}
0 & 1
\end{bmatrix}
\]
\[
\begin{bmatrix}
1.796579e-06 & 3.300672e-06
\end{bmatrix}
\]
\[
\text{$\sigma$}
\]
\[
\text{mad}
\]
\[
\begin{bmatrix}
1.837199
\end{bmatrix}
\]
\[
\text{$p$}
\]
\[
\text{value}
\]
\[
\begin{bmatrix}
0.02
\end{bmatrix}
\]

This function can be used with univariate data (i.e., centroid size) if imported as matrix with row names giving the taxa names,

\[
\text{Csize <- matrix(Y.gpa$Csize, dimnames=list(names(Y.gpa$Csize))) \# make matrix}
\]
\[
\text{Csize with names}
\]
> compare.evol.rates(plethspecies$phy,Csize, gp=gp.end, iter=49)

$sigma.d
[1] 3.013353e-09

$sigmad.all
 0            1
1.776216e-09 5.487627e-09

$sigmad.ratio
[1] 3.089505

$pvalue
[1] 0.38

3.11 Calculate morphological disparity for one or more groups (morphol.disparity)

Function estimates morphological disparity and performs pairwise comparisons to identify differences between groups.

Function:

morphol.disparity(A, groups, iter = 999)

Arguments:

A A 2D array (n x [p x k]) or 3D array (p x k x n) containing GPA-aligned coordinates for a set of specimens
groups A factor defining groups
iter Number of iterations for permutation test

The function takes as input GPA-aligned shape data and a grouping factor, and estimates disparity as the Procrustes variance for each group, which is the sum of the diagonal elements of the group covariance matrix (Zelditch et al. 2012). The group Procrustes variances are used as test values, and these are then statistically evaluated through permutation, where the rows of the shape matrix are randomized relative to the grouping variable. The function can be used to obtain disparity for the whole dataset by using a dummy group factor assigning all specimens to one group, in which case only Procrustes variance is returned.

> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)   #GPA-alignment
> morphol.disparity(Y.gpa$coords, groups=plethodon$site, iter = 99)

$Disp.obs
    ProcVar
Allo 0.001981131
Symp 0.004647113

$Prob.Disp
    Allo   Symp
Allo 1.00 0.01
Symp 0.01 1.00

Use plotTangentSpace() to view the morphospace and color by group.
3.12 Quantify and compare shape change trajectories (trajectory.analysis)

Function estimates attributes of shape change trajectories or motion trajectories for a set of Procrustes-aligned specimens and compares them statistically.

**Function:**

```
trajectory.analysis(f1, data = NULL, estimate.traj = TRUE,
                   traj.pts = NULL, iter = 99)
```

**Arguments:**

- `f1` A formula for the linear model (e.g., `y~x1+x2`), where `y` is a two-dimensional array of shape data
- `data` An optional value specifying a data frame containing all data (not required)
- `estimate.traj` A logical value indicating whether trajectories are estimated from original data; described below
- `iter` Number of iterations for significance testing
- `traj.pts` An optional value specifying the number of points in each trajectory (if `estimate.traj` = `FALSE`)

The function quantifies phenotypic shape change trajectories from a set of specimens, and assesses variation in these parameters via permutation. A shape change trajectory is defined by a sequence of shapes in tangent space. These trajectories can be quantified various attributes (their size, orientation, and shape), and comparisons of these attribute enables the statistical comparison of shape change trajectories (see Adams and Collyer 2007; Collyer and Adams 2007; Adams and Collyer 2009; Collyer and Adams 2013).

Data input is specified by a formula (e.g., `Y~X`), where 'Y' specifies the response variables (trajectory data), and 'X' contains one or more independent variables (discrete or continuous). The response matrix 'Y' must be in the form of a two-dimensional data matrix of dimension (n x [p x k]), rather than a 3D array. The function `two.d.array()` can be used to obtain a two-dimensional data matrix from a 3D array of landmark coordinates. It is assumed that the order of the specimens 'Y' matches the order of specimens in 'X'.

There are two primary modes of analysis through this function. If "estimate.traj=TRUE" the function estimates shape trajectories using the least-squares means for groups, based on a two-factor model (e.g., `Y~A+B+A:B`). Under this implementation, the last factor in 'X' must be the interaction term, and the preceding two factors must be the effects of interest. Covariates may be included in 'X', and must precede the factors of interest (e.g., `Y~cov+A*B`). In this implementation, 'Y' contains a matrix of landmark coordinates. It is assumed that the landmarks have previously been aligned using Generalized Procrustes Analysis (GPA).

If "estimate.traj=FALSE" the trajectories are assembled directly from the set of shapes provided in 'Y'. With this implementation, the user must specify the number of shapes that comprise each trajectory. This approach is useful when the set of shapes forming each trajectory have been quantified directly (e.g., when motion paths are compared: see Adams and Cerney 2007). With this implementation, variation in trajectory size, shape, and orientation are evaluated for each term in 'X' (see Adams and Cerney 2007).

Once the function has performed the analysis, it generates a plot of the trajectories as visualized in the space of principal components (PC1 vs. PC2). The first point in each trajectory is displayed as white, the last point is black, and any middle points on the trajectories are in gray. The colors of trajectories follow the order in which they are found in the dataset, using R's standard color palette: black, red, green, blue, cyan, magenta, yellow, and gray.

Example to estimate trajectories from LS means in 2-factor model

```r
> data(plethodon)
> Y.gpa<-two.d.array(gpagen(plethodon$land)$coords)
> trajectory.analysis(Y.gpa~plethodon$species*plethodon$site,iter=15)
$ProcDist.lm
               df  SS.obs        MS       F  P.val    Rsq
plethodon$species                 1 0.02925784 0.029257838 14.54367 0.0625 0.1485624
```
Compare motion trajectories

```r
> data(motionpaths)
> motionpaths # Motion paths represented by 5 time points per motion

$trajectories

[2,] 3.1034540 11.794735  2.445819 13.830566  3.197470 18.280553  6.432909...

$groups

[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4
Levels: 1 2 3 4

> trajectory.analysis(motionpaths$trajectories~motionpaths$groups, estimate.traj=FALSE, traj.pts=5, iter=15)

$MANOVA.location.covariation

       df   SS.obs        MS       F  P.val       Rsq
motionpaths$groups  3  6520.905 2173.6349 849.846 0.0625 0.9860764
Total              39  6612.981  169.5636      NA     NA        NA

$ANOVA.Size

     Df SumsOfSqs MeanSqs F.Model     R2 Pr(>F)
motionpaths$groups  3   103.345  34.448  51.392 0.8107 0.0625
Residuals          36    24.131   0.670         0.1893
Total              39   127.475  169.5636      NA     NA

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```
### $\text{ANOVA.Dir}$

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>SumsOfSqs</th>
<th>MeanSqs</th>
<th>F.Model</th>
<th>R²</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>motionpaths$\text{groups}$</td>
<td>3</td>
<td>20490.4</td>
<td>6830.1</td>
<td>90.147</td>
<td>0.88252</td>
<td>0.0625</td>
</tr>
<tr>
<td>Residuals</td>
<td>36</td>
<td>2727.6</td>
<td>75.8</td>
<td></td>
<td>0.11748</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>39</td>
<td>23218.0</td>
<td></td>
<td></td>
<td>1.00000</td>
<td></td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

---

Two Dimensional View of Phenotypic Trajectories

![Two Dimensional View of Phenotypic Trajectories](image)
Section 4: Visualization

4.1 Principal Components Analysis (plotTangentSpace)
Function plots a set of Procrustes-aligned specimens in tangent space along their principal axes, i.e., function performs a Principal Component Analysis (PCA).

Function:

```
plotTangentSpace(A, axis1 = 1, axis2 = 2, warpgrids = TRUE, mesh = NULL, 
label = FALSE, groups = NULL, verbose = FALSE)
```

Arguments:

- `A`: A 3D array (p x k x n) containing landmark coordinates for a set of aligned specimens
- `warpgrids`: A logical value indicating whether deformation grids for shapes along X-axis should be displayed
- `mesh`: A mesh3d object to be warped to represent shape deformation along X-axis (when `warpgrids=TRUE`) as described in `warpRefMesh()`.
- `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 X-axis (default = PC2)
- `label`: A logical value indicating whether labels for each specimen should be displayed
- `groups`: An optional factor vector specifying group identity for each specimen
- `verbose`: A logical value indicating whether the output is basic or verbose

The function performs a principal components analysis of shape variation and plots two dimensions of tangent space for a set of Procrustes-aligned specimens (default is PC1 vs. PC2). The percent variation along each PC-axis is returned. Additionally (and optionally, `warpgrids=T`), deformation grids can be requested, which display the shape of specimens at the ends of the range of variability along PC1. If groups are provided, specimens from each group are plotted using distinct colors based on the order in which the groups are found in the dataset, and using R's standard color palette: black, red, green, blue, cyan, magenta, yellow, and gray.

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)   #GPA-alignment
> ref<-mshape(Y.gpa$coords)
> plotTangentSpace(Y.gpa$coords, groups = paste(plethodon$species, plethodon$site))
```

Importance of components:

| PC1     | PC2     | PC3     | PC4     | PC5     | PC6     | PC7     | PC8     | PC9     | PC10    | PC11    | PC12    | PC13    | PC14    | PC15    | PC16    | PC17    | PC18    | PC19    | PC20    | PC21    | PC22    | PC23    | PC24    |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Standard deviation 0.04307 0.03958 0.02035 0.01509 0.01314 0.01293 0.01245 0.01119 0.009202 0.008289 0.007369 0.006651 0.005116 0.004389 0.004173 0.009202 0.008289 0.007369 0.006651 0.005116 0.004389 0.004173 |
| Proportion of Variance 0.36743 0.31023 0.08201 0.04512 0.03418 0.03312 0.03068 0.02478 0.016770 0.013600 0.010750 0.008760 0.005180 0.003810 0.003450 0.016770 0.013600 0.010750 0.008760 0.005180 0.003810 0.003450 |
| Cumulative Proportion 0.36743 0.67767 0.75967 0.80479 0.83897 0.87209 0.90277 0.92755 0.944320 0.957920 0.968670 0.977430 0.982620 0.986430 0.989880 0.993280 0.996360 0.998190 0.999250 1.000000 1.000e+00 1.000e+00 |

<table>
<thead>
<tr>
<th>PC16</th>
<th>PC17</th>
<th>PC18</th>
<th>PC19</th>
<th>PC20</th>
<th>PC21</th>
<th>PC22</th>
<th>PC23</th>
<th>PC24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard deviation 0.004141 0.003944 0.003042 0.002310 0.001955 1.401e-15 2.802e-16 3.426e-17 1.041e-17 0.000e+00 0.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proportion of Variance 0.003400 0.003080 0.001830 0.001060 0.000750 0.000e+00 0.000e+00 0.000e+00 0.000e+00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cumulative Proportion 0.003400 0.003080 0.001830 0.001060 0.000750 0.000e+00 0.000e+00 0.000e+00 0.000e+00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
When verbose=TRUE, the function also returns a list contacting the PC scores ($pc.scores$) and the minimum and maximum shapes ($pc.shapes$) on the two PCs plotted, which can be used in plotRefToTarget().

```r
$pc.scores
PC1       PC2       PC3       PC4       PC5       PC6
[1,] -0.0369931316 0.051182469 -3.128812e-03 -0.0109363280 -0.0264503045
[2,] -0.0007493756 0.059420824 0.0001371746 -2.768676e-03 -0.0081174155
...
```

```r
$pc.shapes
$pc.shapes$PC1min
[,1]         [,2]
[1,]  0.16960517 -0.02807690
[2,]  0.21810341 -0.10340535
...
$pc.shapes$PC1max
[,1]         [,2]
[1,]  0.14235007 -0.020486784
[2,]  0.18546661 -0.066270202
...
$pc.shapes$PC2min
[,1]         [,2]
[1,]  0.13128088 -0.03616776
[2,]  0.18767216 -0.10979356
...
$pc.shapes$PC2max
[,1]         [,2]
[1,]  0.1846661 -0.003640617
[2,]  0.20487416 -0.066270202
...
```
4.2 Plot allometric patterns in landmark data (plotAllometry)

Function performs a regression and plots allometry curves for a set of specimens.

Function:

plotAllometry(A, sz, groups = NULL, method = c("CAC", "RegScore", "PredLine"), warpgrids = TRUE, iter = 99, label = FALSE, mesh = NULL, verbose = FALSE)

Arguments:

A A 3D array (p x k x n) containing landmark coordinates for a set of specimens
sz A vector of centroid size measures for all specimens
groups An optional vector containing group labels for each specimen if available
method Method for estimating allometric shape components; see below for details
warpgrids A logical value indicating whether deformation grids for small and large shapes should be displayed
mesh A mesh3d object to be warped to represent shape deformation of the directional and fluctuating components of asymmetry if warpgrids = TRUE (as described in warpRefMesh()).
iter Number of iterations for significance testing
label A logical value indicating whether labels for each specimen should be displayed
verbose A logical value indicating whether the output is basic or verbose

The function performs a regression of shape on size, and generates a plot that describes the multivariate relationship between size and shape derived from landmark data (i.e., allometry). The abscissa of the plot is log(centroid size) while the ordinate represents shape. Three complementary approaches can be implemented to visualize allometry, "method=CAC", "method=RegScore", "method=PredLine". For all methods, both centroid size and allometry scores are returned. Optionally, deformation grids can be requested, which display the shape of the smallest and largest specimens relative to the average specimen (using 'warpgrids=T' or 'warpgrids=F'). Finally, if groups are provided, the above approaches are implemented while accounting for within-group patterns of covariation (see references for explanation). In this case, the
regression is of the form: shape = size + groups (Note: to examine the interaction term use procD.lm()). Specimens from each group are plotted using distinct colors based on the order in which the groups are found in the dataset, and using R's standard color palette: black, red, green, blue, cyan, magenta, yellow, and gray.

If "method=CAC" (the default) the function calculates the common allometric component of the shape data, which is an estimate of the average allometric trend within groups (Mitteroecker et al. 2004). The function also calculates the residual shape component (RSC) for the data.

```r
> data(ratland)
> Y.gpa <- gpagen(ratland) # GPA-alignment
> plotAllometry(Y.gpa$coords, Y.gpa$Csize, method = "CAC", iter=5)
[1] "No specimen names in data matrix. Assuming specimens in same order."
[1] "No specimen names in size vector. Assuming specimens in same order."
$ProcDist.lm
df SS.obs MS F P.val Rsq
csz 1 0.6280794 0.628079386 474.7793 0.1666667 0.7455947
Total 163 0.8423871 0.005168019 NA NA NA
```

If "method=RegScore" the function calculates shape scores from the regression of shape on size, and plots these versus size (Drake and Klingenberg 2008). For a single group, these shape scores are mathematically identical to the CAC (Adams et al. 2013).

```r
> plotAllometry(Y.gpa$coords, Y.gpa$Csize, method = "RegScore", iter=5)
[1] "No specimen names in data matrix. Assuming specimens in same order."
[1] "No specimen names in size vector. Assuming specimens in same order."
$ProcDist.lm
df SS.obs MS F P.val Rsq
csz 1 0.6280794 0.628079386 474.7793 0.1666667 0.7455947
Total 163 0.8423871 0.005168019 NA NA NA
```
If "method=PredLine" the function calculates predicted values from a regression of shape on size, and plots the first principal component of the predicted values versus size as a stylized graphic of the allometric trend (Adams and Nistri 2010)

```r
> plotAllometry(Y.gpa$coords,Y.gpa$Csize,method="PredLine", iter=5)
[1] "No specimen names in data matrix. Assuming specimens in same order."
[1] "No specimen names in size vector. Assuming specimens in same order."
$ProcDist.lm
   df  SS.obs     MS      F   P.val     Rsq
  csz  1  0.6280794 0.6280794 474.7793 0.1666667 0.7455947
Total 163  0.8423871  0.005168019 NA       NA       NA
```

When verbose=TRUE, the function returns a list containing the ANOVA table ($ProcDist.lm) a matrix of allometry shape scores ($allom.score), a matrix of log centroid size ($LogCsize), a matrix of the predicted shapes for the regression ($pred.shape) – the latter of which can be used with plotRefToTarget(). If "method=CAC" the list also has the residual shape component (RSC) ($resid.shape).

**Protip!** This function is designed to show shape variation with log(centroid size) for allometric study questions. It is however possible to use this function to view shape changes associated with any other continuous variable (e.g., body length, or precipitation). The function automatically takes the natural log of the x-variable. Therefore when using this function for variables that should not be logged, simply input them
as the exponent (exp(x)) and ignore the x-axis graph labels. The allometric shape scores are returned when verbose=T, and these can be used to plot the graph how the user wishes.

4.3 Plot phylogenetic tree and specimens in tangent space (plotGMPhyloMorphoSpace)
Function plots a phylogenetic tree and a set of Procrustes-aligned specimens in tangent space, i.e., a phylomorphospace. This function is also called within physignal().

*Function:*

```r
plotGMPhyloMorphoSpace(phy, A, labels = TRUE, ancStates = TRUE)
```

*Arguments:*

- **phy** A phylogenetic tree of class phylo - see read.tree in library ape
- **A** A matrix (n x [p x k]) or 3D array (p x k x n) containing GPA-aligned coordinates for a set of specimens
- **labels** A logical value indicating whether taxa labels should be included
- **ancStates** A logical value indicating whether ancestral state values should be returned

The function creates a plot of the first two dimensions of tangent space for a set of Procrustes-aligned specimens. The phylogenetic tree for these specimens is superimposed in this plot revealing how shape evolves (e.g., Rohlf 2002; Klingenberg and Gidaszewski 2010). The plot also displays the ancestral states for each node of the phylogenetic tree, whose values can optionally be returned (ancStates = TRUE).

```r
> data(plethspecies)
> Y.gpa<--gpagen(plethspecies$land)    #GPA-alignment
> plotGMPhyloMorphoSpace(plethspecies$phy,Y.gpa$coords)
```

4.4 Create a mesh3d object warped to the mean shape (warpRefMesh)
A function to take a ply file and use thin-plate spline method to warp the file into the estimated mean shape for a set of aligned specimens. *This mesh is used in functions where mesh= option is available.*

*Function:*

```r
warpRefMesh(file, mesh.coord, ref, color = NULL, centered = FALSE)
```

*Arguments:*

- **file** A ply file
- **mesh.coord** GPA coordinates for a set of specimens
- **ref** Reference mesh
- **color** Color for the mesh
- **centered** Whether to center the mesh
Function takes a ply file and its landmark coordinates 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

For landmark coordinates digitized with `geomorph` digitizing functions `centered=TRUE` by default. 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 and `mesh=` option is available (`plotTangentSpace()`, `plotAllometry()`, `bilat.symmetry()`, and `plotRefToTarget()`).

For example,

```r
> #1
> average <- mshape(Y.gpa$coords) # calculate the mean shape

> #2
> findMeanSpec(Y.gpa$coords) # Identify specimen closest to mean shape (input is a 3D array of GPA-aligned coordinates)
specimen7  # returns the name of the specimen
25         # returns the specimen number (where it appears in the 3D array)

> # 3
filelist = c("specimen7.nts","specimen7.nts") # makes false list so a single nts file can be read in using readmulti.nts
> specimen7 <- readmulti.nts(filelist)
  remove(filelist) # read in the nts file, then remove the unnecessary objects

> # 4
> averagemesh <- warpRefMesh("specimen7.ply", specimen7[,1], average, color=NULL)
```

The function returns a mesh3d object of the mesh warped to the shape of the average. It also plots the imported mesh (here “specimen7”) and the new warped mesh:
Now the object “averagemesh” from our example above can be used in `geomorph` functions where shape deformations are plotted (plotTangentSpace(), plotAllometry(), bilat.symmetry(), and plotRefToTarget()).

The “averagemesh” can also be saved to the working directory using the rgl function writePLY().

**findMeanSpec**

*Function:*

```
findMeanSpec(A)
```

A function to identify which specimen lies closest to the estimated mean shape for a set of aligned specimens. A is a 3D array (p x k x n) containing landmark coordinates for a set of aligned specimens. This function is used to facilitate finding a specimen to use with warpRefMesh().

```
> findMeanSpec(Y.gpa$coords)
specimen7   # returns the name of the specimen
25           # returns the specimen number (where it appears in the 3D array)
```

### 4.5 Plot landmark coordinates for all specimens (plotAllSpecimens)

Function plots landmark coordinates for a set of specimens.

*Function:*

```
plotAllSpecimens(A, mean = TRUE, links = NULL, pointscale = 1, meansize = 2)
```

*Arguments:*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A 3D array (p x k x n) containing GPA-aligned coordinates for a set of specimens</td>
</tr>
<tr>
<td>mean</td>
<td>A logical value indicating whether the mean shape should be included in the plot</td>
</tr>
<tr>
<td>links</td>
<td>An optional matrix defining for links between landmarks</td>
</tr>
<tr>
<td>pointscale</td>
<td>An optional value defining the size of the points for all specimens</td>
</tr>
<tr>
<td>meansize</td>
<td>An optional value defining the size of the points representing the average specimen</td>
</tr>
</tbody>
</table>
The function creates a plot of the landmark coordinates for all specimens. This is useful for examining patterns of shape variation after GPA. 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.

Example for 2D data

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)
> plethodon$links
 [,1] [,2] 
[1,] 4 5 
[2,] 3 5 
[3,] 2 4 
[4,] 1 2 
[5,] 1 3 
[6,] 6 7 
[7,] 7 8 
[8,] 8 9 
[9,] 9 10 
[10,] 10 11 
[11,] 11 12 
[12,] 12 1 
[13,] 1 9 
[14,] 1 10
> plotAllSpecimens(Y.gpa$coords,links=plethodon$links)
```

Example for 3D data

```r
> data(scallops)
> gpagen(A=scallops$coorndata, curves=scallops$curvslide, surfaces=scallops$surfslide)
```

**Protip!** Typing the name of the function in the console brings up the function code. The user can look at this code for ideas of how to customize their own graphs.

### 4.6 Plot shape differences between a reference and target specimen (plotRefToTarget)

Function plots shape differences between a reference and target specimen.

**Function:**

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

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"
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
...
Additional parameters to be passed to plot, plot3d or shade3d.

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. This function combines numerous plotting functions found in Claude (2008). Four methods for plots are available, TPS, vector, points, surface.

The function can be used to show shape deformation using shape coordinates from most analyses. Below are examples to visualize shape deformations from the mean shape to another specimen in the dataset.

A 2D data example

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land)  #GPA-alignment
> ref<-mshape(Y.gpa$coords)

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.

> plotRefToTarget(ref,Y.gpa$coords[,39], method="TPS")

> plotRefToTarget(ref,Y.gpa$coords[,39],mag=3, method="TPS")  #magnify difference by 3X

vector: a plot showing the vector displacements between corresponding landmarks in the reference and target specimen is shown.

> plotRefToTarget(ref,Y.gpa$coords[,39],method="vector", mag=3)
```
52

**points** A plot is displayed with the landmarks in the target (black) overlaying those of the reference (gray). 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.

```
> plotRefToTarget(ref,Y.gpa$coords[,39],method="points", mag=3)
```

A 3D data example using the three methods shown above

```
> 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="TPS", mag=3)
```

**X,Y tps grid**

**Y,Z tps grid**
> plotRefToTarget(ref,Y.gpa$coords[,1],method="vector", mag=3)
> plotRefToTarget(ref,Y.gpa$coords[,1],method="points", mag=3)

A mesh3d surface is warped using thin-plate spline (for 3D data only). Requires mesh3d object in option `mesh=`, preferably made using `warpRefMesh()` 4.4, which provides a mesh3d object that is the shape of the sample mean. It is recommended that the mean shape is used as the reference for warping (see Rohlf 1998).

A 3D data example using the average mesh made with `warpRefMesh()` (Note: this example is not included in the `geomorph` package, but show here for illustrative purposes).

> # averagemesh is a mesh made with warpRefMesh, that is the shape of the mean of a set of specimens
> ref <- mshape(Y.gpa$coords) # calculate the mean shape from a set of GPA-aligned specimens
> plotRefToTarget(M1=ref, M2=Y.gpa$coords[,1], mesh=averagemesh, method="surface")

Function plots the warped “target” shape (shown here against the mean for illustrative purposes only).

This function can be used to show deformations between any two sets of coordinates. Coordinate data are provided by several functions, a few examples given below.
Example 1, for plotTangentSpace() verbose=TRUE returns a list containing shape coordinates
($pc.shapes) for the minimum and maximum shape on the two PCs plotted (default is PC1 and PC2) as four
matrices. To use, enter the coordinate matrix into position M2, e.g.,

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land) #GPA-alignment
> ref<-mshape(Y.gpa$coords)
> res <- plotTangentSpace(Y.gpa$coords, groups = paste(plethodon$species,
plethodon$site), verbose=TRUE)
> plotRefToTarget(M1=ref, M2=res$pc.shapes$PC1min, method="TPS") #shape change along
PC1 in the negative direction
> plotRefToTarget(M1=ref, M2=res$pc.shapes$PC1max, method="TPS") #shape change along
PC1 in the positive direction
```

Example 2, for plotAllometry() verbose=TRUE returns a list containing the predicted shape scores
($pred.shape). To use, plot the minimum or maximum of this array in M2, e.g.,

```r
> data(ratland)
> Y.gpa<-gpagen(ratland) #GPA-alignment
> ref<-mshape(Y.gpa$coords)
> res <- plotAllometry(Y.gpa$coords,Y.gpa$Csize,method="RegScore", verbose=TRUE)
> plotRefToTarget(ref, A$pred.shape[,,which.max(Y.gpa$Csize)],method="TPS") # Predicted shape at max centroid size
> plotRefToTarget(ref, A$pred.shape[,,which.min(Y.gpa$Csize)],method="TPS") # Predicted shape at min centroid size
```

Example 3, for two.b.pls(), morphol.integr( method="PLS") and phylo.pls() verbose=TRUE
returns the PLS scores for block 1 and block 2 ($Xscores, $Yscores). M2 plots the actual specimen that is
at the maximum or minimum of block 1. For example, taking the minimum and maximum along the Xblock
1 PLS 1, e.g.,

```r
> res<morphol.integr(Y.gpa$coords[1:5,,], Y.gpa$coords[6:12,,],
method="PLS",iter=99,verbose=TRUE)
> ref<-mshape(Y.gpa$coords)
> plotRefToTarget(ref,Y.gpa$coords[,,which.min(res$x.scores)],method="TPS") #Min
along PLS1
> plotRefToTarget(ref,Y.gpa$coords[,,which.max(res$x.scores)],method="TPS") #Max
along PLS1
```

Example 4, for viewing differences between groups (shown to be significantly different with procD.lm()),
first average the data by groups, calculate group means and plot those means.

```r
> data(plethodon)
> Y.gpa<-gpagen(plethodon$land) #GPA-alignment
> y <- two.d.array(Y.gpa$coords)
> means <- aggregate(y ~ plethodon$site, FUN=mean)
> means
plethodon$site V1   V2  V3   V4   V5     V6      V7
V8 1 Allo 0.1397395 -0.02535584 0.1832314 -0.09279558 -0.02462463 -0.006231783 -0.2797076 -0.09041826
 0.00041826
2 Symp 0.1655262 -0.02134492 0.2056699 -0.09249063 -0.04259982 -0.008460664 -0.2816868 -0.09528204
```

54
4.7 Plot 3D specimen, fixed landmarks and surface semilandmarks (plotspec)

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

Function:

```r
plotspec(spec, digitspec, fixed = NULL, 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).
- `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 plot3d.

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 `digit.surface()` 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`).

```r
> data(scallopPLY)  
> ply <- scallopPLY$ply  
> digitdat <- scallopPLY$coords  
> plotspec(spec=ply, digitspec=digitdat, fixed=16, centered = TRUE)  
```

The fixed landmarks and curve semilandmarks are in red, and the surface semilandmarks are in green.
Section 5: Data Collection (Digitizing)

5.1 2D data collection (digitize2d)
An interactive function to digitize two-dimensional landmarks from .jpg files.

*Function:*

\[
digitize2d(file, nlandmarks, scale, verbose = TRUE)
\]

*Arguments:*

- **file**: Name of jpeg file to be digitized. File names can be written in manually, including paths, or obtained using directory/file manipulation functions e.g., list.files
- **nlandmarks**: Number of landmarks to be digitized.
- **scale**: Length of scale placed in image.
- **verbose**: User decides whether to digitize in verbose or silent format (see details), default is verbose

Function for digitizing 2D landmarks on specimen images (.jpg). "nlandmarks" is the number of landmark points to be digitized by the user. Landmarks should include "true" landmarks and semi-landmarks to be "sliders". For best results, digitizing sequence should proceed by selecting all true landmark points first, followed by selection of sliding semi-landmarks. Use function define.sliders.2d to select sliding semi-landmarks.

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). If "y", the user is asked to continue to select the next landmark. If "n", the user is asked to select it again. This can be repeated until the user is comfortable with the landmark chosen. verbose=FALSE is silent, and digitizing of each landmark is continuous and uninterrupted until all landmarks are chosen.

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

Digitize the scale bar by selecting the two end points (single click for start and end),

\[
> \text{digitize2d("mvz206608l.jpg", nlandmarks=11, scale=3, verbose = FALSE)}
\]

Set scale = 3

![Digitize scale bar](image)

Digitize each landmark with single click and the landmark is shown in red.

Select landmarks 1:11
When selection of n landmarks is completed, an *.nts file is created in working directory using the specimen name, adding *.nts as a suffix.

```
mvz206608l.nts
"mvz206608l
1 11 2 0 dim=2
0.913794820018337 0.728605670332276
1.04745503254995 0.586844883885935
0.654575013896415 0.765058455568172
0.168537877417812 0.671901337743106
0.103732925887331 0.736706289273586
0.0551292122394711 0.813662169216032
0.103732925887331 0.927070834394373
0.330550256244013 1.04452980904337
0.686977489661655 1.00402671433682
1.03530410413799 0.927070834394373
1.71575609520803 0.643549171448521
```

Landmark coordinates are returned scaled.

### 5.2 Define sliding semilandmarks in 2D (define.sliders.2d)

An interactive function to define which landmarks will "slide" along two-dimensional curves.

**Function:**

```plaintext```
define.sliders.2d(spec, nsliders)
```

**Arguments:**

- `spec` Name of specimen, as an object matrix containing 2D landmark coordinates
- `nsliders` Number of landmarks to be semilandmarks that slide along curves

Function takes a matrix of digitized landmark coordinates and helps user choose which landmarks will be treated as "curve 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.
> data(hummingbirds)
> define.sliders.2d(hummingbirds$land[,1], nsliders=10)

**Selection:** 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.

![Example of landmark selection](image1)

![Example when finished](image2)
This procedure is overlapping, so for example, a curve defined by a sequence of semilandmarks, the user must select the 2nd point of the first three to be the 1st for the next e.g., 1 2 3 then 2 3 4, then 3 4 5 etc. Function returns a .csv file to the working directory.

```
curveslide.csv
before slide after
 1    11    12
11    12    13
13    14    15
 7    15    14
12    13    14
 1    16    17
16    17    18
17    18    19
18    19    20
10    20    19
 2    21    22
21    22    23
22    23    24
23    24    25
 8    25    24
```

which can be read in for use with gpagen()

```r
> curves <- as.matrix(read.csv("curveslide.csv", header=T))
```

### 5.3 Importing 3D surface files (read.ply)
A function to read ply files, which can be used for digitizing landmark coordinates or for shape warps. Other 3D file formats are currently not supported. Other files can be converted to ply using for example Meshlab (http://meshlab.sourceforge.net).

**Function:**

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

**Arguments:**

- `file` An ASCII ply file
- `ShowSpecimen` A logical value indicating whether or not the ply file should be displayed

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 function opens the ply file and plots the mesh, with faces rendered if file contains faces, and colored if the file contains vertex color.

```r
> # usage, reading a file called myply.ply in the working directory
> read.ply("myply.ply", ShowSpecimen=TRUE)

> # view an example in the geomorprph package
> data(scallopPLY)
> myply <- scallopPLY$ply
> attributes(myply)
$names
 [1] "vb" "it" "primitivetype" "material"
```

$class
> plot3d(myply)

> # change color of mesh
> myply$material <- "gray" # using color word
> myply$material <- "#FCE6C9" # using RGB code

5.4 3D data collection: landmarks (digit.fixed)
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().

Function:

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 Numeric The number landmarks (fixed, and curve sliders if desired)
index Logical Whether selected landmark addresses should be returned
Function for digitizing "n" three-dimensional landmarks. The landmarks are "fixed" (traditional landmarks). They can be later designated as "curve sliders" (semilandmarks, that will "slide" along curves lacking known landmarks if required. A sliding semi-landmark ("sliders") will slide between two designated points, along a line tangent to the specified curvature, and must be defined as "sliders" using function define.sliders.3d() or with similar format matrix made outside R. For 3D "surface sliders" (surface semilandmarks that slide over a surface) the function digitsurface() should be used instead.

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: 3D Digitizing functions in geomorph are interactive between landmark selection using a mouse (see below for instructions), and the R console. 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):

- the RIGHT mouse button (primary) to select points to be digitized (click-drag a box around a vertex to select as landmark)
- the LEFT mouse button (secondary) is used to rotate mesh,
- 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:

- press button to rotate 3D mesh,
- press button while pressing COMMAND key to select points to be digitized (click-drag a box around a vertex to select as landmark),
- press button while pressing OPTION key to adjust mesh perspective,
- the mouse SCROLLER or trackpad two finger scroll is used to zoom in and out.

XQuartz must be configured: go to Preferences > Input > tick “Emulate three button mouse”.

NOTE: there is no pan (translate) functionality in rgl library for all platforms at this time. This is why the function has a center=TRUE/FALSE option.

Example, reading in a mesh with 24700 vertices,

```
> mandible <- read.ply("Mandible.ply")
> digit.fixed(mandible, fixed=20, index=F, ptsize=1, center=T)
Select Landmark 1 #
Keep landmark 1(y/n)? #See picture step 2
y
Select Landmark 2 #See picture step 2
Keep landmark 2(y/n)?
y
Select Landmark 3 #See picture step 1
```
Function writes to the working directory an NTS file with the name of the specimen and .nts suffix containing the landmark coordinates.

```
mandible.nts
"mandible
1 20 3 0 dim=3
13.2628089578878 10.2361171749175 -3.49646546534654
9.32980895788779 18.7309171749175 -1.94616546534654
-2.70691104211221 17.5351171749175 0.852234534653466
-13.7911610421122 15.6363171749175 0.945934534653464
-14.1681010421122 6.40941717491749 0.853034534653467
...
```

If `index=FALSE` (default) 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 containing a matrix containing the x,y,z coordinates of the digitized landmarks ($selected$) and a matrix of addresses for landmarks that are "fixed" ($fix$, primarily for internal use).

## 5.5 3D data collection: landmarks and semilandmarks

**buildtemplate**

An interactive function to build template of three-dimensional 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()`.

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

**Arguments:**

- `spec` Name of surface file, as either an object of class shape3d/mesh3d, or matrix of three-dimensional vertex coordinates.
- `fixed` The number of fixed template landmarks
Function constructs a template of fixed landmarks and n "surface sliders", semilandmarks that slide over a surface. The user digitizes the fixed points, then the function finds n surface semilandmarks following algorithm outlined in Gunz et al. (2005) and Mitteroecker and Gunz (2009). Surface semilandmarks are roughly equidistant set of predetermined number of points, chosen over the mesh automatically using a nearest-neighbor approach. The set of fixed and surface slider landmarks are exported as a "template", which is used to extract a set of similarly numbered landmarks on every specimen using function `digitsurface()`. Some of the "fixed" landmarks can be later designated as "curve sliders" using function `define.sliders.3d()` if required - see details in `digit.fixed()`.

To ensure a strong match between the scan and the template, it is recommended that a reasonable number of fixed points be used. These fixed points can be designated as "curve sliders" later using function `define.sliders.3d`, see the function `digit.fixed()` for details. 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:** as before in `digit.fixed()`.

Example, digitizing 16 landmarks on a mesh and then automatically calculate 150 surface sliding semilandmarks

```r
> myply1 <- read.ply("myply1.ply", ShowSpecimen=FALSE)
> buildtemplate(m ply1, 16, 150, ptsize=1)
Select Landmark 1 #See picture step 1
Keep landmark 1(y/n)? #See picture step 2
y
```

![Image of digitizing process](image.png)
Select all of the fixed landmarks, and then the function will calculate the position of the surface sliding landmarks on the mesh, and plot them in blue.

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 \texttt{digitSurface()}, and "surfslide.csv", a file containing the address of the landmarks defined as "surface sliders" for use with \texttt{gpagen()}. Function also returns to console an \(n \times 3\) matrix containing the \(x,y,z\) coordinates of the digitized landmarks.

\begin{verbatim}
myply1.nts
"myply1
1 166 3 0 dim=3
13.2628089578878 10.2361171749175 -3.49646546534654
9.32980895788779 18.7309171749175 -1.94616546534654
-2.70691104211221 17.5351171749175 0.852234534653466
-13.7911610421122 15.6363171749174 0.945934534653464
-14.1681010421122 6.40941717491749 0.853034534653467
...
\end{verbatim}

\begin{verbatim}
template.txt
"xpts" "ypts" "zpts"
13.2628089578878 10.2361171749175 -3.49646546534654
9.32980895788779 18.7309171749175 -1.94616546534654
-2.70691104211221 17.5351171749175 0.852234534653466
-13.7911610421122 15.6363171749174 0.945934534653464
-14.1681010421122 6.40941717491749 0.853034534653467
...
\end{verbatim}
which can be read in for use with gpagen()

> sliders <- as.matrix(read.csv("surfslide.csv", header=T))

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

Function:

editTemplate(template, fixed, n)

Arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template</td>
<td>Matrix of template 3D coordinates.</td>
</tr>
<tr>
<td>fixed</td>
<td>Number of &quot;fixed&quot; landmark points (non surface sliding points)</td>
</tr>
<tr>
<td>n</td>
<td>Number of points to be removed</td>
</tr>
</tbody>
</table>

Function edits a 'template.txt' file made by buildtemplate(), which must be in current working directory. Function overwrites 'template.txt' in working directory with edited version. Use read.table("template.txt", header = T).

> template <- as.matrix(read.table("template.txt", header=T, sep=" "))
> editTemplate(template, fixed=16, n=2)

Remove Template Points
1 of 2 points have been removed
2 of 2 points have been removed # right click and drag box around landmark
digitsurface

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().

Function:

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

Arguments:

spec Name of surface file, as either an object of class shape3d/mesh3d, or matrix of three-dimensional vertex coordinates.
fixed numeric: The number of fixed template landmarks
ptsise numeric: Size to plot the mesh points (vertices), e.g., 0.1 for dense meshes, 3 for sparse meshes
center Logical Whether the object 'spec' should be centered prior to digitizing (default center=TRUE)

Function for digitizing fixed 3D landmarks and placing "surface sliders", semilandmarks that slide over a surface. Following selection of fixed points (see digitizing below), function finds surface semilandmarks following algorithm outlined in Gunz et al. (2005) and Mitteroecker and Gunz (2009). digitsurface finds the same number of surface semilandmarks as the “template.txt” file (created by buildtemplate()) by downsampling scanned mesh, registering template with current specimen via GPA. A nearest neighbor algorithm is used to match template surface landmarks to current specimen's. To use function digitsurface, the template must be constructed first, and “template.txt” be in the working directory.

Some of the "fixed" landmarks digitized with digitsurface can be later designated as "curve sliders" using function define.sliders.3d if required (see details in digit.fixed). 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: as before in digit.fixed().

Example, digitizing 16 landmarks on a mesh and fit a template to add 150 surface sliding semilandmarks

```r
> myply2 <- read.ply("myply2.ply", ShowSpecimen=FALSE)
> digitsurface(myply2, fixed=16, pysize=1
  Select Landmark 1
  Keep Landmark 1(y/n)?
  y
  ...

Select all of the fixed landmarks, and then the function will calculate the position of the surface sliding landmarks on the mesh based on the template (blue), and plot them in green.
```

![Image of digitized landmarks on a mesh]
Function writes to the working directory an NTS file with the name of the specimen and .nts suffix containing the landmark coordinates.

```
myply2.nts
"myply2
1 166 3 0 dim=3
13.6450089578878 9.85741717491749 -3.49156546534654
9.70490895788779 18.3504171749175 -1.90256546534653
-2.33683104211221 17.5362171749175 0.853034534653467
-14.1518710421122 15.2515171749175 1.06203453465347
-14.1681010421122 6.40941717491749 0.853034534653467
...
```

5.6 Define sliding semilandmarks in 3D (define.sliders.3d)
An interactive function to define which digitized landmarks of an '*.nts' file will "slide" along three-dimensional (3D) curves.

Function:
```
define.sliders.3d(spec, nsliders, surfsliders = FALSE)
```
Arguments:
- `spec` Name of specimen, as an object matrix containing 3D landmark coordinates
- `nsliders` Number of landmarks to be semilandmarks that slide along curves
- `surfsliders` If `spec` contains landmarks that are "surface sliders", made by `buildtemplate()`, "surfslide.csv" should be in working directory

Function takes a matrix of digitized landmark coordinates, such as made by `digit.fixed()` or `buildtemplate()` and helps user choose which landmarks will be treated as "curve 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.

To define the sliders, for each sliding landmark along the curve in the format 'before-slider-after':
1) Click-drag box around landmark to choose the first landmark between which semi-landmark will "slide",
2) Click-drag box to choose sliding landmark,
3) Click-drag box 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.

```
> define.sliders.3d(16,11)
```
semilandmark 9 slides between landmarks 5 and 10

This procedure is overlapping, so for example a curve defined by a sequence of semilandmarks, the user must select the 2nd point of the first three to be the 1st for the next e.g., 1 2 3 then 2 3 4, etc.

Function returns a 'curves x 3' matrix containing the landmark address of the curve sliders, indicating the points between which the selected point will "slide", written to the working directory as "curveslide.csv".

```
curveslide.csv
before  slide after
5   9   10
9  10    7
10  7   11
  7  11   12
11 12    6
12  6   13
  6 13   14
13 14    8
14  8   15
  8 15   16
15 16    1
```

which can be read in for use with `gpagen()`

```r
> curves <- as.matrix(read.csv("curveslide.csv", header=T))
```
Frequently Asked Questions

1) I’ve loaded my TPS file but I get a warning that no names were extracted. My TPS file has specimen names in it, what do I do?

   - For TPS files it is necessary to specify whether the specimen name should be read from the “ID=” line or the “IMAGE=” line. See section 1.1 for more details.

2) I want to add a group to my data analysis, what do I do?

   - Grouping variables can be made in R (e.g., c( )) or imported from outside (e.g., as a .csv file). See section 2.4 for more details.

3) How many iterations should I use in permutation tests?

   - The default for geomorph functions is 999. Generally more is better, but too many can be excessive and does not lead to more power. See section on permutation tests in the introduction for more details.

4) How do I obtain allometry residuals for further analyses in geomorph?

   - In short, you don’t. Instead, include size as a covariate. Reason: MANCOVA is correct, MANOVA on residuals may not be (if different slopes then residuals from common slope not correct, or if one uses residuals on each group separately then they are no longer comparable).

5) I’m using digit.fixed/buildtemplate/digitsurface. When I choose a landmark, the mesh image disappears. And I get this error: Error in rgl.user2window(x, y, z, projection = proj) : NA/NaN/Inf in foreign function call (arg 5). What am I doing wrong?

   - For 3D digitizing in R with geomorph, one must click and drag a square around the vertex to select it. See section 5.4 for more details and a demonstration.
References


