Quick Guide to Geomorph

Emma Sherratt

10 February 2016

Getting Started

I. 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/. Occasionally, we make updates between uploads to CRAN. Users can install via GitHub the current beta version from https://github.com/geomorphR/geomorph.

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

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

```
##
## To cite package 'geomorph' in a publication use:
##
##   Adams, D.C., and E. Otarola-Castillo. 2013. geomorph: an R
##   package for the collection and analysis of geometric
##   morphometric shape data. Methods in Ecology and Evolution.
##   4:393-399.
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {geomorph: an R package for the collection and analysis of geometric morphometric shape
##     author = {D.C. Adams and E. Otarola-Castillo},
##     journal = {Methods in Ecology and Evolution},
##     year = {2013},
##     volume = {4},
##     pages = {393-399},
##   }
##
## As geomorph is evolving quickly, you may want to cite also its
## version number (found with 'library(help = geomorph)').
```

Also, since the package is quickly evolving, it is advisable to cite the CRAN package and version as:

Adams, D.C., M. L. Collyer, and E. Sherratt. 2016 *geomorph*: Software for geometric morphometric analyses. *R* package version 3.x http://cran.r-project.org/web/ packages/geomorph/index.html.

II. 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.

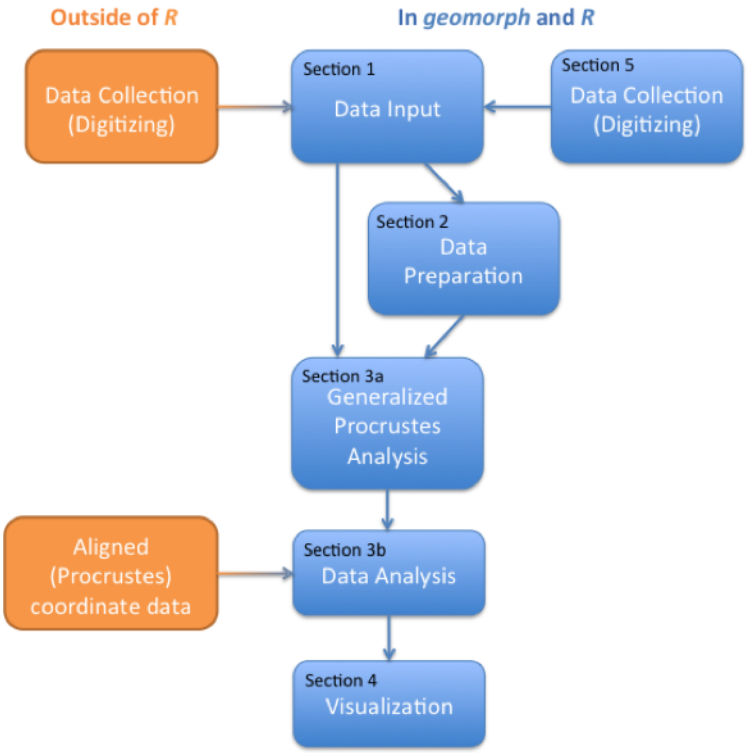


Figure 1 Overview of the morphometric analysis process. In blue are the steps performed in *R* and *geomorph*, and those in orange are done outside of *R* and imported in.

In **Vignette 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 **Vignette 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 **Vignette 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 **Vignette 3b**, the statistical analysis functions are presented in order by type of analysis (**Table 1**). In **Vignette 4**, we describe how to plot and visualize the data analysis results, including shape deformation graphs and ordination plots (e.g., PCA). In **Vignette 5** the functions that can be used to generate coordinate data from 2D images and 3D surface files (i.e., an ASCII .ply) are discussed. And finally, at the end there are some frequently asked questions and their solutions.

Table 1 Functions in *geomorph*.

Input	Preparation	Analysis
read.morphologika	arrayspecs	advanced.procD.lm
readland.nts	define.links	bilat.symmetry
readland.tps	define.modules	compare.evol.rates
readmulti.nts	estimate.missing	compare.multi.evol.rates
	findMeanSpec	globalIntegration
	fixed.angle	gpagen
	mshape	integration.test
	two.d.array	morphol.disparity
	writeland.tps	phylo.integration
		phylo.modularity
		physignal
		procD.lm
		procD.pgls
		trajectory.analysis
		two.b.pls

Visualisation	Datasets	Digitizing
procD.allometry	hummingbirds	buildtemplate
gridPar	mosquito	define.sliders
plotAllSpecimens	motionpaths	digit.curves
plotGMPhyloMorphoSpace	plethodon	digit.fixed
plotOutliers	plethspecies	digitize2d
plotRefToTarget	plethShapeFood	digitsurface
plotspec	pupfish	editTemplate
plotTangentSpace	ratland	read.ply
warpRefMesh	scallopPLY	
warpRefOutline	scallops	
	larvalTails	

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
`code` 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:

```
procD.allometry(f1, f2 = NULL, f3 = NULL, logsz = TRUE, iter = 999,
  seed = NULL, alpha = 0.05, RRPP = TRUE, data = NULL, ...)

readland.tps(file, specID = c("None", "ID", "imageID"), readcurves = FALSE,
  warnmsg = TRUE)
```

In dark blue, the function name and options. In black, an object, usually data, a fomula of data, or sometimes a file name. In green, a multipart option, requires choice of ONE of the presented values. In brown, a logical

option that requires a TRUE or FALSE input, or an option that requires a value. In blue, an option that requires a numeric 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"`.

Finally, I occasionally write source code for very specific issues to complement *geomorph* functions. They can be found here: http://emsherratt.github.io/MorphometricSupportCode/.

III. Installing *geomorph* and *R*

These instructions assume you already have *R* (and perhaps also RStudio) installed.

To install *geomorph* from CRAN

```
install.packages("geomorph", dependencies = TRUE)
```

This will install the latest version of *geomorph* from CRAN https://cran.rstudio.com/.

Alternatively, if you prefer menus:
Rapp: Packages & Data > Package Installer > Choose CRAN (binaries) from drop down menu, type *geomorph* in box and click get list. Select *geomorph*, select install dependencies box, and click install selected.
or
Rstudio: Packages tab > Install: Install from CRAN repository > type *geomorph* in box and select install dependencies box, and click install.

Installing from GitHub

CRAN restricts the number of updates package maintainers can make in a year. Occasionally, bugs slip through that need to be fixed immediately. We maintain a "Stable" version of the current CRAN version of *geomorph* in our GitHub repository, which can be installed as source.

To install the source package from GitHub:

```
install.packages("devtools", dependencies = TRUE)
devtools::install_github("geomorphR/geomorph",ref = "Stable")
```

Installing the beta version

We have a beta version for the upcoming version that contains the most current updates and new features. It is held on a GitHub repository:

To install the beta *geomorph* package:

```
devtools::install_github("geomorphR/geomorph",ref = "Develop")
```

Installing compilers for Mac users: Previous versions of *geomorph* required users to have compilers installed in order to install packages from source. This is no longer necessary from *geomorph* versin 3.0. However the information is provided here if any issues arise.

1) Go to the Mac App store and download Xcode Development Tools https://itunes.apple.com/au/app/xcode/id497799835?mt=12 and follow install instructions. For OS10.6, follow the instructions here http://kitcambridge.tumblr.com/post/17778742499/installing-the-xcode-command-line-tools-on-snow.

2) Download the compilers. For OS10.8 and below: go to CRAN website here and download the GNU Fortran compiler (gfortran-4.2.3.pkg) and follow install instructions. For OS10.9 and above: Open Terminal (Applications/Utilities) and type in:

```
curl -O http://r.research.att.com/libs/gfortran-4.8.2-darwin13.tar.bz2
```

This will download the installer. Then type

```
sudo tar fvxz gfortran-4.8.2-darwin13.tar.bz2 -C /
```

This will install the compilers into the /usr/local/lib/ folder. The command sudo will ask for your password. Type it in, but note it will not appear on the line. Press return. The terminal window will fill with all the files being written. (This information is thanks to the The Coatless Professor http://r.research.att.com/libs/gfortran-4.8.2-darwin13.tar.bz2 sudo tar fvxz gfortran-4.8.2-darwin13.tar.bz2 -C /).

3) Download and install XQuartz (X11) http://xquartz.macosforge.org/trac if it is not already on your Mac. It will be installed in the Utilities folder. This program must be running every time you use *geomorph* (required by rgl).

This is a short version of information available here http://cran.r-project.org/bin/macosx/tools/ and here http://r.research.att.com/tools/.

Installing compilers for Windows users: Go to the *R* website and download RTools http://cran.rstudio.com/bin/windows/Rtools. Make sure to download the correct version! Follow the install instructions. You may need to modify the path (asked during the installing). Try first without ticking the box on the install window. If running `install_github()` above does not work then reinstall the RTools with the new path box ticked). Alternatively, use the function `find_rtools()` in devtools package.

For more information for Windows users, see here http://cran.r-project.org/doc/manuals/R-admin.html#The-Windows-toolset.

Using *geomorph*

Regardless of how you install *geomorph*, in order to use it you must start every session by loading the package

```
library(geomorph)
```

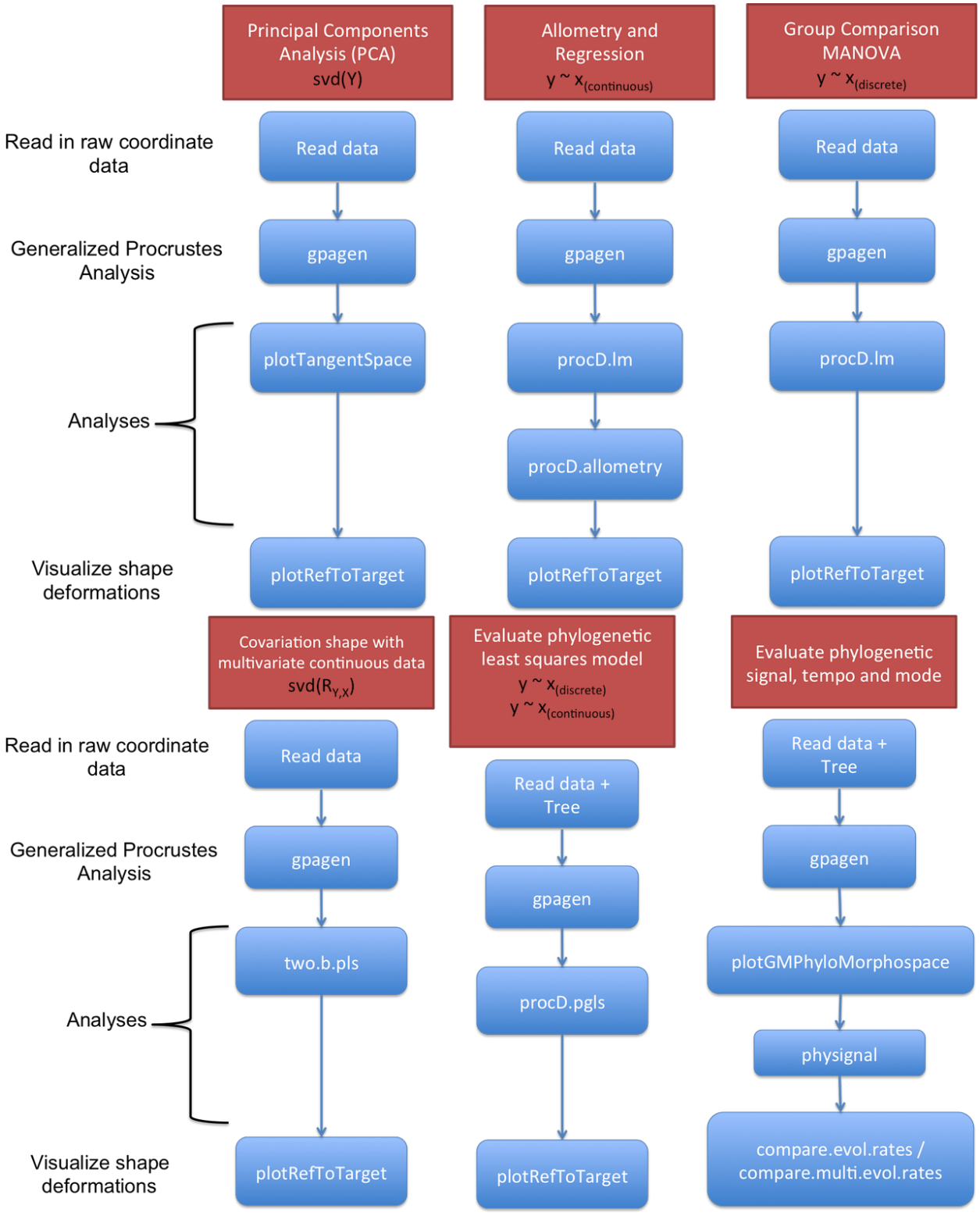
```
## Loading required package: rgl
```

```
## Loading required package: ape
```

You'll notice that a black warning message is printed in the console saying the package rgl and ape are also loaded. All of the 3D plots of interactive functions of *geomorph* are run through rgl https://cran.r-project.org/web/packages/rgl/index.html. ape is called for several phylogenetic analyses.

IV. 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.



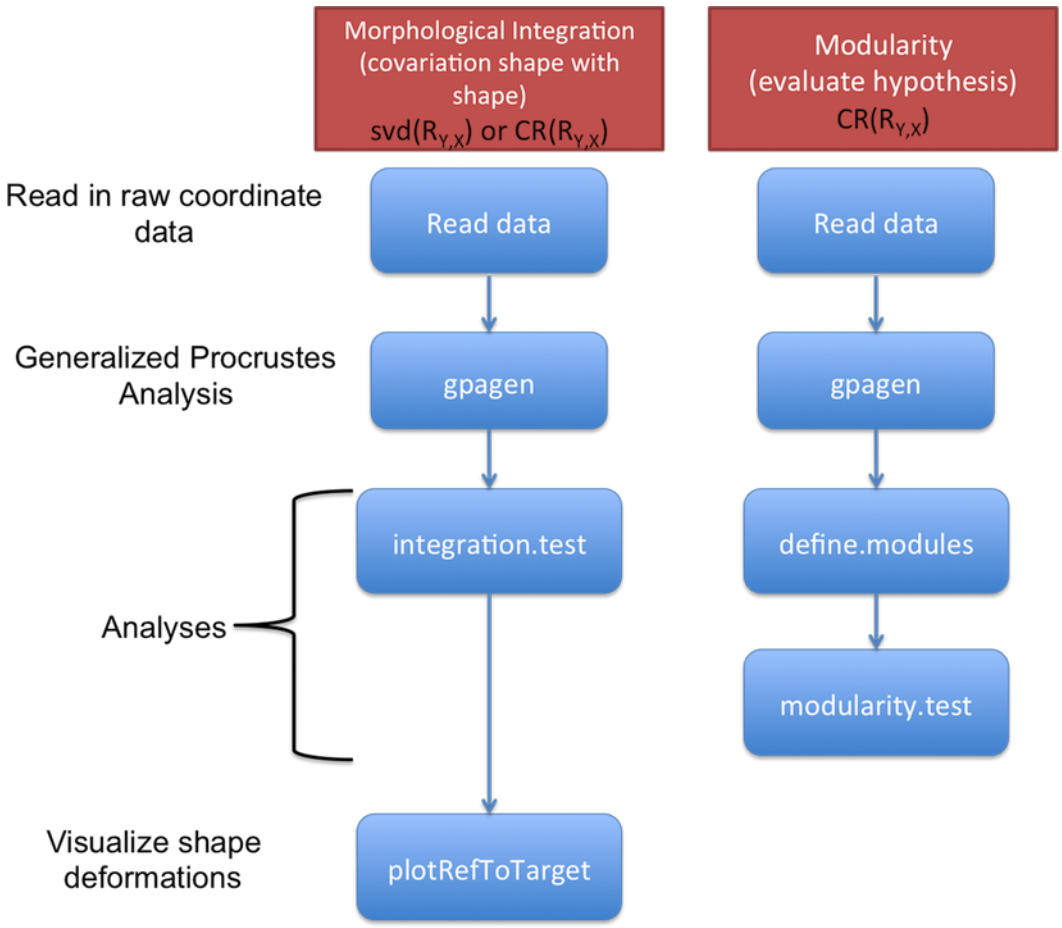


Figure 2 Example workflows in geomorph of morphometric analyses

V. Example datasets in *geomorph*

For vignettes 2 through 4, many of the examples will be using data included with *geomorph*. There are ten datasets: `plethodon`, `scallops`, `hummingbirds`, `larvalTails`, `mosquito`, `ratland`, `plethspecies`, `plethShapeFood`, `motionpaths` and `scallopPLY`. 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.
To load an example dataset:

```
data(plethodon)
attributes(plethodon)
```

```
$names
[1] "land"    "links"   "species" "site"    "outline"
```

```
head(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
[5,] 1.28180 53.18484
[6,] 1.24236 53.32288
```

```
head(plethodon$links)
```

```
     [,1] [,2]
[1,]    4    5
[2,]    3    5
[3,]    2    4
[4,]    1    2
[5,]    1    3
[6,]    6    7
```

```
plethodon$species
```

```
 [1] Jord  Jord  Jord  Jord  Jord  Jord  Jord  Jord  Jord  Jord  Teyah
[12] Teyah Teyah Teyah Teyah Teyah Teyah Teyah Teyah Teyah Jord  Jord
[23] Jord  Jord  Jord  Jord  Jord  Jord  Jord  Jord  Teyah Teyah Teyah
[34] Teyah Teyah Teyah Teyah Teyah Teyah Teyah
Levels: Jord Teyah
```

```
plethodon$site
```

```
 [1] Symp Symp Symp Symp Symp Symp Symp Symp Symp Symp Symp Symp Symp Symp Symp
[15] Symp Symp Symp Symp Symp Symp Allo Allo Allo Allo Allo Allo Allo Allo
[29] Allo Allo Allo Allo Allo Allo Allo Allo Allo Allo Allo Allo Allo
Levels: Allo Symp
```

```
head(plethodon$outline)
```

```
          [,1]       [,2]
[1,] 0.3986487 -0.2022766
[2,] 0.4000364 -0.2022312
[3,] 0.4014240 -0.2021857
[4,] 0.4028116 -0.2021400
[5,] 0.4041990 -0.2020943
[6,] 0.4055864 -0.2020484
```

The dataset above, `plethodon`, is a `list` containing several components: the coordinate data (`plethodon$land`), two sets of grouping variables as factors (`plethodon$species`, `plethodon$site`), the wirelink addresses (`plethodon$links`), and an matrix of outline coordinates for visualizations (`plethodon$outline`).

```
class(plethodon$site)
```

```
[1] "factor"
```

```
class(plethodon$land)
```

```
[1] "array"
```

```
class(plethodon$links)
```

```
[1] "matrix"
```

Note that your own data will not necessary be a list - depending on how you read in each element of your data to *R*. However, learning how to manipulate the eaxmple datasets here will give you practive for working with lists later on.

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
```

geomorph primarily has data stored in a 2D array or 3D array (matrix and array respectively) (see below and 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

Data arrays

Landmark data in *geomorph* can be found as objects in two formats: a 2D array (matrix; Figure 3A) or a 3D array (Figure 3B). These data formats follow the convention in other morphometric packages (e.g., shapes, Morpho) and in J.Claude's book Morphometrics in *R* (2008), and help to distinguish Shape Variables from other continuous morphometric data (linear measurements).

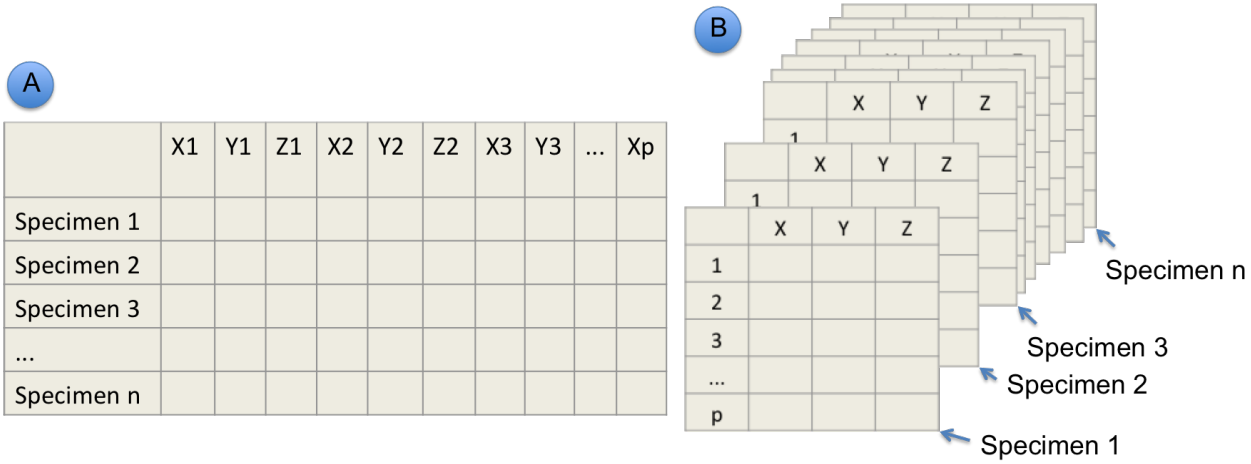


Figure 3 2D array and 3D array of landmark coordinate data. NOTE: This example shows 3D landmark coordinate data, but the same format would be used for 2D coordinate data.

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(plethodon$land)
```

```
[1] 12  2 40
```

If `dim()` gives three numbers, it is a 3D array. Here mydata has p=12, k=2, n=40. If `dim()` gives two numbers, 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).

```
dim(two.d.array(plethodon$land))
```

```
[1] 40 24
```

***geomorph* data frame**

A data frame in *R* is usually used for storing tables data tables. In *geomorph* a `geomorph.data.frame` is a special list that contains all data to be used in your analyses. The purpose is similar to the base *R* function, `data.frame`, but without the constraint that data must conform to an `n` (observations) x `p` (variables) matrix. Rather, the list produced is constrained only by `n`. List objects can be Procrustes residuals (coordinates) arrays, matrices, variables, distance matrices, and phylogenetic trees. Results from `gpagen()` can be directly imported into a geomorph.data.frame to utilize the coordinates and centroid size as variables. The `geomorph.data.frame` is of particular importance when using the functions: `procD.lm()`, `procD.pgls()`, `advanced.procD.lm()`, `morphol.disparity()`, `trajectory.analysis()`.

VI. 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. In several functions (e.g., `procD.lm`, `procD.gls`, `advanced.procD.lm`, `procD.allometry`), two possible resampling procedures are provided. First, if `RRPP=FALSE`, the rows of the matrix of shape variables are randomized relative to the design matrix. This is analogous to a 'full' randomization. Second, if `RRPP=TRUE`, a residual randomization permutation procedure is utilized (Collyer et al. 2014). Here, residual shape values from a reduced model are obtained, and are randomized with respect to the linear model under consideration. These are then added to predicted values from the remaining effects to obtain pseudo-values from which SS are calculated. NOTE: for single-factor designs, the two approaches are identical. However, when evaluating factorial models it has been shown that RRPP attains higher statistical power and thus has greater ability to identify patterns in data should they be present (see Anderson and terBraak 2003).

Statistical Designs

For functions requiring a linear model formula, `f1`, the following is a guide for different models:

An expression of the form y ~ model is interpreted as a specification that the response y is modelled by a linear predictor specified symbolically by model.

Common designs	f1
Simple Linear Regression	`y ~ x`
Single-factor MANOVA	`y ~ a`
Single-factor MANCOVA	`y ~ x * a` [means `a + x + a:x`, where : denotes interaction]
Multiple-factor MANOVA	`y ~ a + b`
Factorial MANOVA	`y ~ a * b`
Nested MANOVA	`y ~ a / b`

For models that have 3 or more factors the option `int.first = TRUE/FALSE` is important. TRUE adds the interactions of first main effects before the subsequent main effects. FALSE adds them in order, for example, the model: `shape ~ a*b*c`
`int.first = FALSE : shape ~ a + b + c + a:b + a:c + b:c + a:b:c`
`int.first = TRUE : shape ~ a + b + a:b + c + a:c + b:c + a:b:c`

For one or two factors, this is inconsequential and thus `int.first` can be left at default.

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/), and the *R* Introduction manual on http://cran.r-project.org/doc/manuals/R-intro.pdf, and various Springer eBooks in the series 'Use *R*!'. Also highly recommended is J. Claude's book Morphometric with *R* (2008).