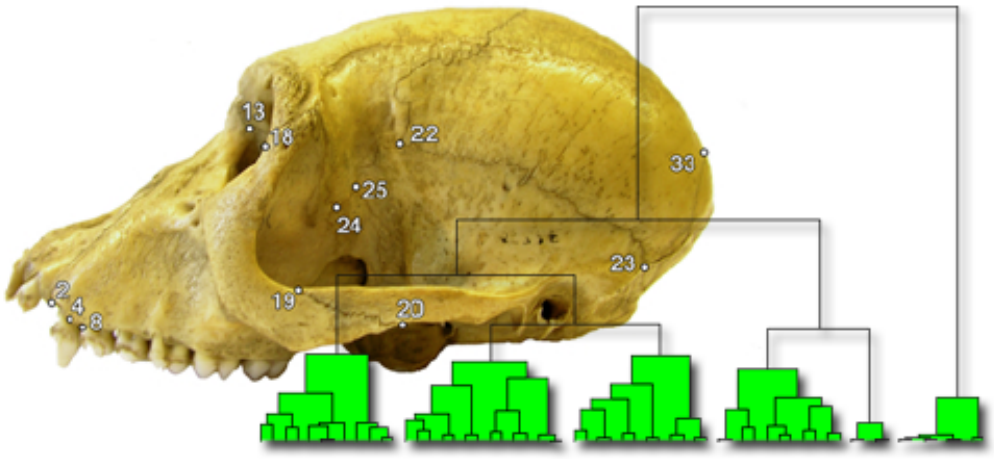
Modularity for Mathematica
User's Guide
Version 2.0

For use with:

Goswami, A. & Polly, P. D. 2010 Methods for studying morphological integration, modularity and covariance evolution. In *Quantitative Methods in Paleobiology. Paleontological Society Short Course, October 30th, 2010*, vol. The Paleontological Society Papers (ed. J. Alroy & G. Hunt), pp. 213-243. Chicago: The Paleontological Society.



P. David Polly
Earth & Atmospheric Sciences
Indiana University
Bloomington, Indiana 47405 USA
https://pollylab.indiana.edu
pdpolly@indiana.edu

Anjali Goswami
The Natural History Museum
Cromwell Road, London SW7 5BD UK
https://goswamilab.com
a.goswami@nhm.ac.uk

Table of Contents

About Modularity and this Package

Morphological integration is correlation of parts, the integration of morphological traits or features that must function, grow, or be passed to offspring on as working units. Individual integrated units are modules, a group of traits that are highly correlated among themselves but only loosely correlated with traits in other modules. The study of integration and modularity was first developed by Olson and Miller (1958) and expanded on by Cheverud (1982), Zelditch (1988), Wagner (1995), Raff (1996), Klingenberg (2000), Mitteroecker (2007) and many others.

The basic component of studying modularity and integration is identifying packages of intercorrelated traits that behave independently of other such packages, whether through ontogeny, across individuals within a population, or during the course of evolution. This package performs a simple analysis of modularity on geometric morphometric landmark data.

This package accompanies the review by Goswami and Polly (2010). Users are referred to that paper for a discussion of the methods embedded in this package, including their strengths and weaknesses, and for elaboration of other methods that address broader problems.

Updates in 2.0

This package was updated 7 April 2018 to fix inconsistencies with the latest versions of Mathematica. No substantial changes were made to functions. Updates include changing *Agglomerate[]* to *DirectAgglomerate[]* for functions that calculate dendrograms from correlation matrices and no longer loading the Multivariate Statistics package to use the *PrincipalComponents[]* function.

Installing the Modularity for Mathematica package

1. Download the latest version of the package at http://mypage.iu.edu/~pdpolly/Software.html (right click on link to save as file)
2. Open the file in Mathematica
3. Under the File menu, choose “Install”
4. Under Type of Item choose “Package”, under source choose the file you just saved, under Install Name enter “Modularity”.
5. Once installed, enter the command “<<Modularity`” to use the functions.

Using Mathematica

Mathematica is a software package for mathematical and statistical analysis from Wolfram Research (http://www.wolfram.com/).

Mathematica has a unique interface that takes a while to get used to. You open to a blank page, like a word processor, where you can type anything you want. Most of the time you will type commands that do things with your data: connect to databases, plot graphs, or carry out calculations. Unlike other statistical or mathematical programs, the commands you type and the output you get remain on the page, which gives you a record of what you've done step-by-step. To help organize your work, you can add headers, format boxes, etc. with the Format Menu.

Cells are an important organizing feature of *Mathematica*. Note the "cell" markers on the right margin. Each cell is bounded by a bracket and commands within a cell are executed together. You can open and close cells by double clicking the bracket. This can be useful if you have lots of stuff in a notebook… you can give a section a heading, which causes cells in that section to be grouped, after which you can close the section by double clicking.

Shift + Enter causes a cell to be executed. Pressing the enter key creates a new line, just like in a word processor, but when you want to execute a command you typed, you type SHIFT+ENTER somewhere in the cell and all commands in the cell are executed.

Mathematica Commands. Mathematic is designed to be as easy to learn as possible so that you can concentrate on working instead of the program. Almost all commands are English words written out in full with capitals at the beginning of words and brackets [] at the end of the command. For example, the command to calculate an average of a set of numbers is *Mean[]*, the command to do a principal components analysis is *PrincipalComponents[]*, and the command to take the logarithm of a number is *Log[]*.

Graphics. Mathematic is good at graphics. You can either use simple functions like *ListPlot[]* to create a generic graph, or you can experiment with *Graphics[]* to create a completely customized graphic.

Example Datasets

Two example datasets are provided with this package. The first set found in the file "*Marmota_siberica.csv*", contains 13 two-dimensional landmarks from the mandibles of a geographically localized subspecies of marmot, *Marmota siberica* (data from Caumul & Polly 2005, sample L, n=14) (Figure 1). These data are arranged in a comma-delimited file with a

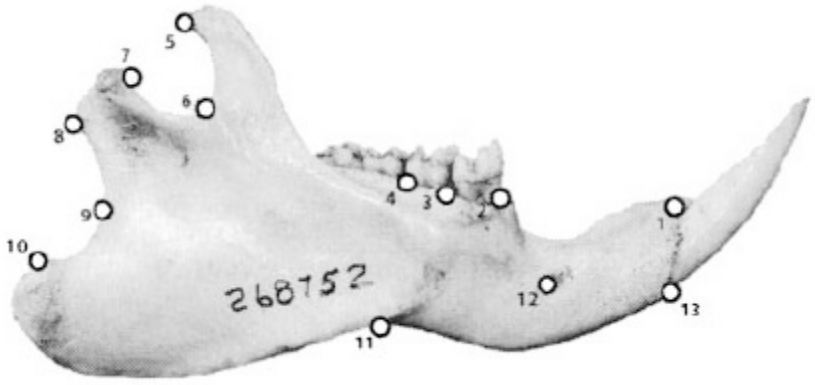


Figure 1. *Mandible landmarks. (from Caumul & Polly 2005).*

specimen identifier in the first column, followed by the x and y coordinate points of the 13 landmarks in the next 26 columns. These data can be loaded into Mathematica and the first column discarded with the following command:

landmarks = Import["**C:\\Marmota_siberica.csv"**, "CSV"][[1;;,2;;]];

where the data are stored in the variable *landmarks*, where the first argument should be the complete path to the data file, and the numbers in double square brackets at the end specify that rows 1 to the end and columns 2 to the end be retained.

The second data set, found in the file "*Macaca_mulata.csv*", contains 61 three-dimensional landmarks from the skulls of a population of Japanese Macaques (*n*=45). This data set is described and illustrated in Goswami and Polly (2010). These data have the x, y and z coordinates of each landmark in the 183 columns, with no identifier. They can be read into Mathematica with the following command:

landmarks = Import["**C:\\Macaca_mulata.csv"**, "CSV"];

Stand-Alone Functions

Procrustes[*landmarks, nlands, ndims*]

The *Procrustes[]* functions superimposes landmarks using the algorithm for generalized Procrustes superimposition presented by Rohlf and Slice (1990). *Procrustes[]* and other geometric morphometric are also found in the PollyMorphometrics package for Mathematica available at http://mypage.iu.edu/~pdpolly/Software.html .

> ***landmarks*** is a two-dimensional matrix with landmark coordinates in columns and specimens in rows.
> ***nlands*** is the number of landmarks.
> ***ndims*** is the number of landmark dimensions (2 or 3).

The following command superimposes the 13 marmot landmarks if they have been loaded as described above and saves the superimposed coordinates as array named *superimposed*:

```
superimposed = Procrustes[landmarks, 13, 2];
```

CongruenceCoefficient[*superimposed*]

The *CongruenceCoefficient[]* function calculates a correlation matrix for two- or three-dimensional landmarks using the congruence coefficient. The congruence coefficient is one of several methods for calculating a single correlation between two multidimensional variables, such as 2D or 3D landmarks (see discussion in Goswami & Polly 2010).

> ***superimposed*** is a three-dimensional array in which the landmarks are arranged with specimens in the first dimension, landmarks in the second dimension, and coordinates for each landmark in the third dimension.

The following command reformats the marmot landmarks from the two-dimensional array produced by *Procrustes[]* into the three-dimensional array required by *CongruenceCoefficient[]*, where 2 is the number of landmarks dimensions and 13 is the number of landmarks:

```
superimposed = Partition[Partition[Flatten[superimposed],2],13];
```

The following command then calculates the 13x13 correlation matrix among the landmarks and saves it in the variable *R*:

```
R=CongruenceCoefficient[superimposed];
```

CanonicalCorrelation[*superimposed*]

The *CanonicalCorrelation[]* function calculates a correlation matrix for two- or three-dimensional landmarks using canonical correlation, one of several methods for calculating a single correlation between two multidimensional variables, such as 2D or 3D landmarks (see discussion in Goswami & Polly 2010).

> ***superimposed*** is a three-dimensional array in which the landmarks are arranged with specimens in the first dimension, landmarks in the second dimension, and coordinates for each landmark in the third dimension.

The following command reformats the marmot landmarks from the two-dimensional array produced by *Procrustes[]* into the three-dimensional array required by *CanonicalCorrelation[]*, where 2 is the number of landmarks dimensions and 13 is the number of landmarks:

```
superimposed = Partition[Partition[Flatten[superimposed],2],13];
```

The following command then calculates the 13x13 correlation matrix among the landmarks and saves it in the variable *R*:

```
R=CanonicalCorrelation[superimposed];
```

Mantel[*A, B, n*]

The *Mantel[]* function does a Mantel test for correlation in two matrices by calculating their matrix correlation and performing a Monte Carlo permutation test to test the significance of the correlation. The function returns three values: the matrix correlation, the proportion times the real matrix correlation was greater than the *n* random permutations, and the *P*-value that the correlation is greater than expected for random matrices.

> ***A*** and ***B*** are two symmetric correlation matrices.
> ***n*** is the number of permutations for the Monte Carlo test of significance. 10000 permutations are recommended.

The following set of commands performs a Mantel test on the correlation matrices from the Marmot data set based on the Congruence Coefficient and Canonical Correlation.

```
superimposed = Procrustes[marmots, 13, 2];

P = CongruenceCoefficient[superimposed];

Q = CanonicalCorrelation[superimposed];

Mantel[P, Q, 10000]
```

The results of that code are

```
{0.789719, 0.9998, 0.0001}
```

,

indicating that the matrix correlation between the two matrices is 0.79, that the correlation is greater than random 99.9% of the time (out of 10,000 permutations) and that the *P*-value for the correlation being greater than random is significant at *P*= 0.0001.

Subsample[*data, nrows*]

The *Subsample[]* function randomly samples rows of a data matrix (e.g., specimens from a data matrix of landmarks).

> ***data*** is the matrix of data to be subsampled.
> ***nrows*** is the number of randomly sampled rows to be returned.

The following code randomly samples five specimens from the Marmot datamatrix:

RandomMarmots=Subsample[marmots, 5];

SubsampleMatrix[*data1,data2, nlands, ndims, m, t*]

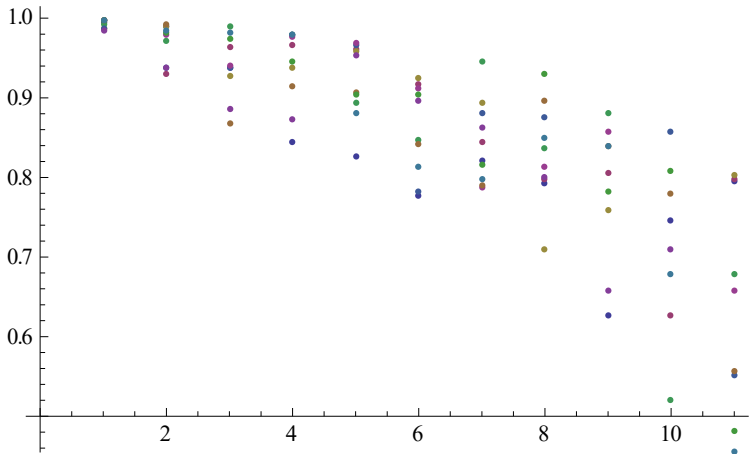
The *SubsampleMatrix[]* performs a subsample analysis to explore the effects of small sample size on matrix correlation for one or two sets of landmark data.

data1 is the matrix of landmark data to be subsampled.
data2 is the matrix of data to which the subsamples are to be compared (this is often the same matrix as *data1* if the purpose is to see how small sample size affects the estimation of a correlation matrix)
nlands is the number of landmarks in the data sets.
ndims is the number of dimensions in each landmark (2 or 3).
m is the maximum number of rows to remove (this should never be larger than *n-3*, where *n* is the total number of rows).
t is the number of trials at each subsample size.

The following code does a subsample analysis on the marmot data, iteratively dropping the sample so that the smallest samples have only three rows and subsampling ten times at each iteration.

rare=Subsample[marmots, marmots, 13, 3, Length[marmots]-3, 10];

Plotting the transpose of the results shows how the correlation between the subsample and the original data deteriorate as the sample becomes smaller:



The correlation of the subsample and the original is near 1.0 at the first step, when only one row has been removed. The correlation drops to 0.5-0.9 at the 11th step when eleven rows have been dropped from the original sample.

Modularity Analysis Functions

Modularity[*landmarks, nlands, ndims, labels*]

The *Modularity[]* function takes a set of unsuperimposed landmarks and does a simple but complete analysis of modularity. For each landmark data set, the function displays a scree plot of the eigenvalues of the correlation matrix and a Ward's dendrogram that shows the similarity of correlations among the landmarks. The function also reports the sample standard deviation of the eigenvalues, their relative sample standard deviation, and the expectation of the relative SD for random, non-modular data. An optional randomization test can be used to estimate the number of modules based on the eigenvalue scree.

> ***landmarks*** is a two-dimensional array of landmarks with coordinates in the columns and specimens in the rows. Landmarks do not need to be superimposed since the function invokes *Procrustes[]* to superimpose them.
> ***nlands*** is the number of landmarks.
> ***ndims*** is the number of landmark dimensions (2 or 3).
> ***labels*** is an array of labels for the landmarks. These will be used to identify the tips of the cluster analysis dendrogram.

The following command creates a numbered list 1 to 13 to be used as labels for the marmot data set:

labels =Table[x, {x, 13};

The following command does a modularity analysis on the marmot landmarks:

Modularity[landmarks, 13, 2, labels]

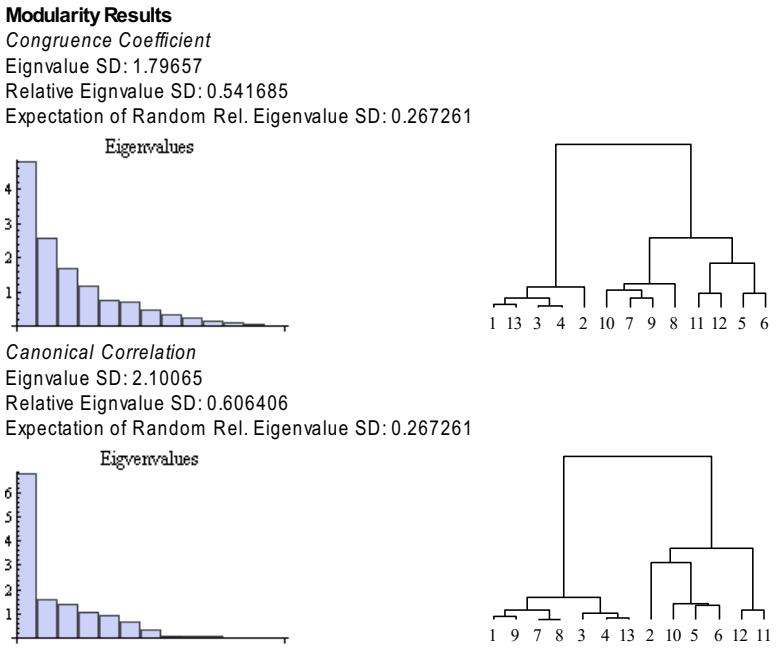


Figure 2*. Modularity analysis results for the marmot mandible data set.*

The program first superimposes the landmarks then creates a correlation matrix using both the congruence correlation and canonical correlation. The eigenvalues of each correlation matrix are calculated and shown as scree plots, along with their sample standard deviation. The relative standard deviation is calculated following Pavlicev et al. (2009) as SD/Sqrt[N-1], where SD is the sample standard deviation of the eigenvalues and N is the number of eigenvalues. The expectation of the relative standard deviation for a random data set is reported for comparison. If the first few relative eigenvalues are especially high and their relative standard deviation is larger than the expectation, then the landmarks have a modular structure (Goswami & Polly 2010; Pavlicev et al. 2009). A Ward's linkage dendrogram is shown for each correlation matrix. Landmarks are clustered according to the strength of their correlations (technically speaking, the correlation matrix is converted to a distance matrix with the transformation 1-Abs[R]). Modules are expected to be clustered in the dendrogram (see discussion of pros and cons in Goswami & Polly 2010).

An optional randomization test is available to statistically test for modularity using the eigenvalue structure of the correlation matrix. When this test is invoked, the landmarks are randomized 100 times and the eigenvalues calculated from the associated correlation matrices to provide a distribution of eigenstructure for data with no modules. The number of significantly correlated modules is estimated as the number of eigenvalues that are higher than the maximum corresponding eigenvalue from the randomized data set. The following command runs the modularity analysis for the same dataset but with the randomization test invoked [note: the randomization process can take several minutes for large data sets]:

Modularity[landmarks, 13, 2, labels, "RandomizationTest"]

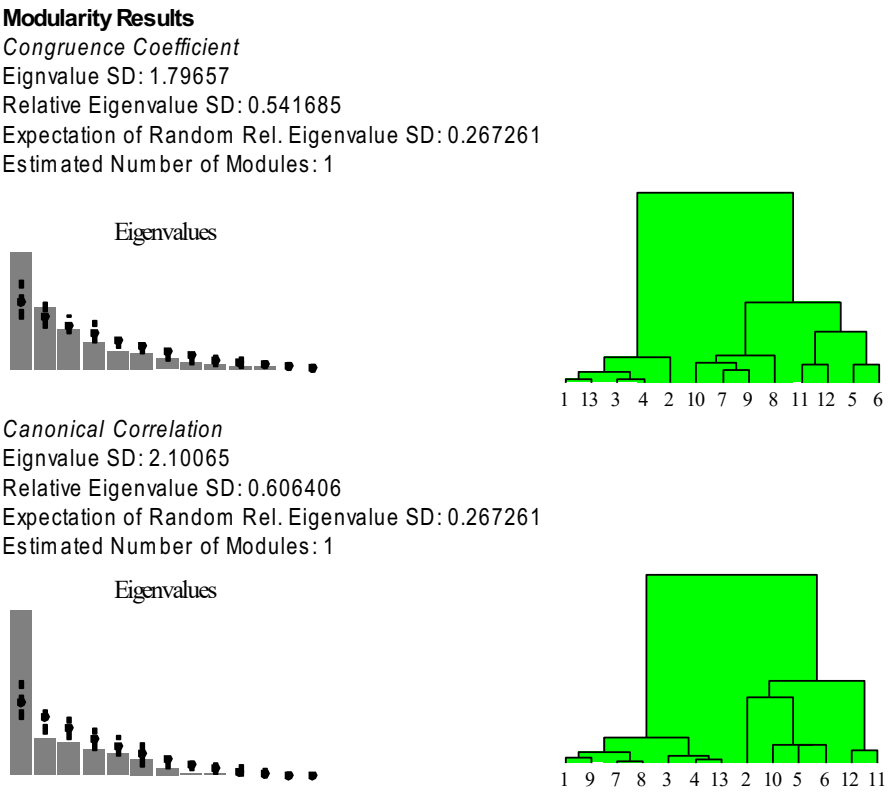


Figure 3. *Modularity results for the marmot mandible data set when the randomization test option is invoked.*

The scree plots now show the eigenvalues of the two correlation matrices as thick grey bars with the randomization results shown as black whisker bars (mean, maximum and minimum). The latter are a guide to the expectation of eigenvalues for completely random data with no real modularity. The number of eigenvalues greater than that expectation is the number of modules with significant structure. This number is used to highlight the modules in the dendrogram. For the marmot mandible data, only one significant module is found regardless of which correlation coefficient is used, suggesting that there is significant integration in this data set, but no modularity. The following output is from the Macaque data set, where several modules were identified. Note that the number of modules may be mis-estimated if there are many eigenvalues that are effectively zero. In some cases the actual value of those tiny eigenvalues exceeds the randomized value and is counted as though it were a module. Visual inspection of the results will show whether this is the case.

Modularity Results
Congruence Coefficient
Eignvalue SD: 1.69774
Relative Eigenvalue SD: 0.219177
Expectation of Random Rel. Eigenvalue SD: 0.149071
Estimated Number of Modules: 6

Canonical Correlation
Eignvalue SD: 2.55376
Relative Eigenvalue SD: 0.384993
Expectation of Random Rel. Eigenvalue SD: 0.149071
Estimated Number of Modules: 6

Figure 4. *Modularity results for the Macaque skull data set with the randomization option invoked.*

Magwene[*R*, *n*, *labels*, criterion, threshold]

The Magwene[] function performs a graphical model analysis as described by Magwene (2001).

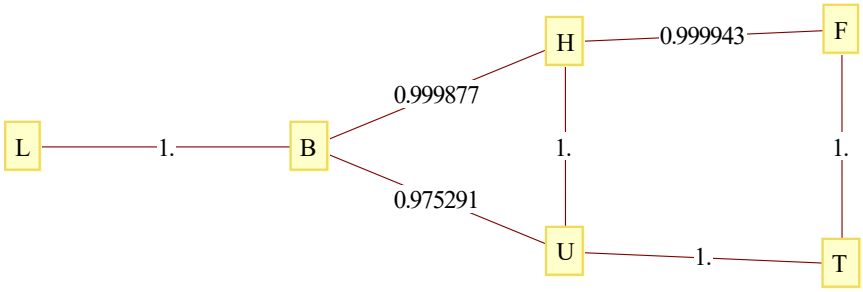
> ***R*** is a correlation matrix.
> ***n*** is the sample size for the data set used to generate the correlation matrix.
> ***labels*** is an array of labels for the variables in the correlation matrix.

There are two optional variables:

> ***Criterion*** specifies which matrix to use for edge labels. Default is p-value for the edge exclusion deviance matrix. Other criteria are "EED" for the edge exclusion deviance value itself, "ES" for edge strength, and "PC" for the partial correlation matrix.
> ***Threshold*** specifies the threshold value used to draw the graph edge (the default value is 0.95 or higher).

The function returns a graphical model for integration and modularity, the partial R-square values for each variable with respect to all others, the matrix of partial correlations among variables, the edge exclusion deviance matrix, the matrix of p-values for edge exclusion deviance, and the edge strength matrix.

The following is the output of the Magwene function for the data presented in his 2001 paper. Note that the implementation of the function in this package does not work if the correlation matrix is singular, which is always the case for landmark data (see discussion in Goswami & Polly 2010):



{RSquare -> , {0.471035, 0.434852, 0.900214, 0.904006, 0.878004, 0.889951}}

References

Caumul, R. & Polly, P. D. 2005 Phylogenetic and environmental components of morphological variation: Skull, mandible, and molar shape in marmots (Marmota, Rodentia). *Evolution* **59**, 2460-2472.

Cheverud, J. 1982 Phenotypic, genetic, and environmental morphological integration in the cranium. *Evolution* **36**, 499-516.

Goswami, A. & Polly, P. D. 2010 Methods for studying morphological integration, modularity and covariance evolution. In *Quantitative Methods in Paleobiology. Paleontological Society Short Course, October 30th, 2010*, vol. The Paleontological Society Papers (ed. J. Alroy & G. Hunt), pp. 213-243. Chicago: The Paleontological Society.

Klingenberg, C. P. & Zaklan, S. D. 2000 Morphological integration between developmental compartments in the Drosophila wing. *Evolution* **53**, 358-375.

Magwene, P. M. 2001 New tools for studying integration and modularity. *Evolution* **55**, 1734-1745.

Mitteroecker, P. & Bookstein, F. 2007 The conceptual and statistical relationship between modularity and morphological integration. *Systematic Biology* **56**, 818-836.

Olson, E. C. & Miller, R. L. 1958 *Morphological Integration*. Chicago: University of Chicago Press.

Pavlicev, M., Cheverud, J. M. & Wagner, G. P. 2009 Measuring Morphological Integration Using Eigenvalue Variance. *Evolutionary Biology* **36**, 157-170.

Raff, R. 1996 *The shape of life: genes, development, and the evolution of animal form*: University of Chicago Press Chicago.

Rohlf, F. J. & Slice, D. 1990 Extension of the Procrustes method for the optimal superimposition of landmarks. *Systematic Zoology* **39**, 40-59.

Wagner, G. P. 1995 Adaptation and the modular design of organisms. *Advances in Artificial Life* **929**, 317-328.

Zelditch, M. L. 1988 Ontogenetic variation in patterns of phenotypic integration in the laboratory rat. *Evolution* **42**, 28-41.