A Practical Companion to Geometric Morphometrics for Biologists:

Running analyses in freely-available software

Miriam Leah Zelditch, Donald L. Swiderski, and H. David Sheets

1

Introduction to the Workbook

The purpose of this workbook is to cover the practical details of a morphometric analysis. The workbook is separate from the textbook because we want to be able to update the workbook more regularly than we revise the textbook. The software for geometric morphometrics changes more rapidly than the theory does. The textbook and workbook, however, are closely linked to each other even if they are now physically separate. Each workbook chapter explains how to do what the corresponding textbook chapter teaches. For example, the third chapter of the textbook covers superimposition methods and the third chapter of the workbook covers the software for superimposition. The one exception to the general rule that the workbook chapter covers the same subject as the textbook chapter with the same number is Chapter 5. In the textbook, Chapter 5 is on the thin-plate spline, which is used primarily for depicting deformed grids so in the workbook, Chapter 5 covers graphics, summarizing the graphical options of various programs, how they can be modified and how the figures can be saved to a file.

We expect that most readers will read the first nine chapters of the textbook (and workbook) because these cover general subjects. We do *not* expect that most readers will read all of the last five chapters because these focus on topics relevant to a particular discipline. Therefore, although we do not regularly repeat the material covered in the first ten chapters, we do repeat material that arises first in the more specialized chapters.

Software options

In this book, we discuss only freely available software. Some programs are free if you are a member of a scientific society sponsoring the software, such as the European Virtual Anthropology Network (EVAN). As a member of that society, you can get compiled software (executables for Windows, Linux and MACs) of the EVAN Toolbox plus documentation and tutorials and a test data set and templates for analyses. If you are *not* a member, you get free access to the source code, to compile on your own. There are also functions that are free but run under (very expensive) packages, such as Mathematica or Matlab.

Your institution may have a site license, making that package free for you, but we do not assume that all readers have free access to either. For purposes of this book (and our own research) we consider only the freely available programs.

Most of the software used in geometric morphometrics can be obtained at the Stony Brook website (life.bio.sunysb.edu/morph/). To see what software is available, go to the Stony Brook home page, and click on "Software" (the second link). There you will see a menu of choices, including "Data acquisition", "Shape coordinates", "Superimposition", "Thin-plate spline," "Multivariate analysis," "Utility programs", "Comprehensive software", "Software for R" and "3D software". Some can be downloaded from the Stony Brook web site, for others you will be redirected to the site from which you can download the software by following the instructions you will find when you click on the link. It is worth periodically checking the Stony Brook site for software updates; these are usually announced on the morphometrics listserve, Morphmet, but not all are announced there, so to avoid missing some that are useful for you, check the site regularly. To receive those announcements, or to follow questions and replies that are posted there, you can find out how to subscribe to Morphmet in the "Other sites" link on the Stony Brook home page.

Before you finish this chapter, you will have downloaded and installed several of these programs, learned how to load the data files, and seen the datafile formats that various programs read. The programs that you will need (or, more precisely, can run) depend partly on your operating system and partly on whether you have two- or three-dimensional data. Relatively few programs can run under both Windows and Mac operating systems and can also analyze both 2- and 3D data. **MorphoJ** written by Chris Klingenberg (2011) is an exception because it is platform independent and can analyze 2- and 3D data. Similarly, the **shapes** package written by Ian Dryden (2012), and the **R** functions published by Julien Claude (2008) in his book *Morphometrics with R* can analyze both 2- and 3D data and they run on any operating system that can run **R** (R_Development_Core_Team, 2012). The other packages are not as versatile. Some packages, such as **IMP** written by David Sheets (e.g., 2011a, 2010, 2011d, 2009, 2011c, 2011b) have versions compiled to run under Windows plus versions compiled to run under the Mac operating system and versions for analyzing 2D data plus versions for 3D data. Other packages, such as the **tps** programs written by Jim Rohlf (e.g., 2006, 2010a, 2010b, 2011), run only under Windows and analyze only 2D data. These programs do run on any Mac equipped to run Windows, however.

The programs that we emphasize in this workbook are within the category of "Comprehensive software" on the Stony Brook website. This category includes two kinds of packages. One is comprehensive in the sense that a single program conducts many analyses—you open the program, load your datafile(s) and select the methods that you wish to use. **MorphoJ** is comprehensive in this sense, as is **PAST (Paleontological Statistics)** (Hammer et al., 2001) and the **shapes** package plus the **R** functions by Claude. The other kind is comprehensive in the sense that there are several related programs, each of which does one kind of analysis (e.g., principal components analysis, regression) but taken together, they comprise an integrated series of programs. The programs within a single series can read the same datafiles and have the same or similar interfaces plus a consistent organization of options and a common array of graphical outputs. When using this class of programs, you open the one that does the analysis that you want to run and load your data. If you have several analyses to do, you might have several programs open at once or you might open one then another and then the next. This class includes the **tps** series and the **IMP** series. Programs in **R** could be regarded as comprehensive in this sense because there are a large number of packages that you can use in R, although you may find yourself occasionally reformatting your data. A unique feature of **R** is that anyone, including you, can write code, which makes this a truly comprehensive package for morphometric analyses. Several functions useful for doing morphometrics in **R** are available at the Stony Brook website (follow the link to "Software for R"), others are published or distributed as supplemental materials in published papers and we include a file of functions (**functionsR**) as well as scripts (**scriptsR**) with this workbook. The scripts include lines that you run interactively and will require some editing for your data sets. We hope that people will add their functions to the ones now available on the Stony Brook website.

Considering that there are multiple programs, and multiple ones that will run on your operating system and analyze your data, whatever its dimensionality, the question is whether the programs are interchangeable. Certainly most can do standard morphometrics analyses, including principal components analysis (sometimes called a "relative warp analysis" in geometric morphometrics), simple statistical analyses, such as linear regression and multivariate t-tests (Hotelling's T^2), and most can also do something that is less standard than it should be: partial least squares. Usually, the programs give virtually the same numerical results (they might differ by a rotation or rounding error). But the programs may nonetheless differ in something important to you because they vary in their options, including the statistical tests they perform or how they conduct the tests, and also in their graphical output.

The most important difference among the comprehensive packages lies in what else they do aside from those standard methods. The programs are not interchangeable because none of them does everything that the others do. Each package has gaps. For example, only two packages have programs that can superimpose semilandmarks, **tpsRelw** (tps) and **CoordGen7**, (IMP); only two can trace shape evolution on a cladogram, **tpsTree** (tps) and **MorphoJ**; only two can do a canonical variates analysis, **CVAGen** (IMP) and **MorphoJ**. Also, several analyses can be done in only one package. As a result, you will likely need multiple packages to conduct a single study. You will likely also need **R** because some methods, most notably, the complex statistical models discussed in Chapter 9 and some methods discussed in Chapters 10-14, are not implemented in *any* specialized morphometrics package. **R** can be intimidating at first, but the alternatives are even *less* friendly (and vastly more expensive) statistical packages. One advantage of doing your analyses in **R** is that you can do complex series of analyses, repeat them as often as you wish by rerunning a script, modify it and rerun the whole series of now modified analyses, keeping a complete record of everything that you did in the script, plus you can tailor both your statistical tests and your graphical output to your own preferences.

In this workbook, we focus on one specialized morphometrics package, the Integrated Morphometrics Package (**IMP**) because one of us, Dave Sheets, writes it. We are therefore most familiar with its capabilities—not only do we know what the programs do, we also know how they do it and where all the options are located on the interface. We regularly use other packages, but cannot promise authoritative coverage of them because we, like other users, must rely on their manuals plus trial-and-error. We do discuss them, but more briefly, and only if we understand how to run the analysis in that package. Most chapters also include a section on doing the analyses in **R**, especially the later chapters that discuss some methods that can be implemented only in **R**. We expect that the sections on doing morphometrics in **R** will need especially frequent updates, so we have a version of the workbook that includes only the sections on **R**, the **R Workbook**. This will be updated on its own schedule. We would note that most of the **R** functions and scripts supplied with the workbook, in **functionsR** and **scriptsR**, were not written by us. When you use them you should cite their authors, *not* us. We would also note that the scripts included within this book and the **R Workbook** have been copied into a Word document. We have tried to ensure that they copied properly but we cannot guarantee that no straight quote was ever transformed into a smart quote or that other special characters were never introduced into the text. Do not try to run the scripts or functions by copying them from these two documents—use the **functionsR** and **scriptsR** instead.

We recommend that you try working with several programs, and begin the process of learning them before you are anxious to get your own results. Whether a program seems easy and friendly to use is largely a matter of familiarity. If you are unfamiliar with a program, everything, including (or especially) loading the data, may seem dauntingly complex. Being familiar with several programs has the obvious advantage that you will not restrict your analyses to the methods available in one package. We recommend downloading several packages, or even all of them.

Getting and installing software

Getting the software:

Go to the Stony Brook website and follow the link to "Software." Whatever you will do with your data, the first task is to collect it. So the first programs to get are the ones for digitizing. Follow the link to "Data acquisition" software. For 2D data, you will find **tpsDig**; there are two versions of the program; get the newest (**tpsDIG2)** but we also recommend getting the older one as well because it is easier to use for some purposes (see the next chapter in this workbook). **tpsDig** is widely used and this is the program that we assume you are using in our next chapter on data acquisition. **tpsDig** however only runs under Windows. A platform-independent program is **ImageJ**, written in Java, available at http://rsbweb.nih.gov/ij/. You should also look at the plug-ins for **ImageJ**, classified by their functions, such as "segmentation." For 3D data, you will find a link to **Landmark Editor** (http://graphics.idav.ucdavis.edu/research/projects/EvoMorph/), which runs under Windows. Download and install the program that you want to use for digitizing. If you have digital images, and can run Windows programs, we recommend **tpsDig**.

Next, if you are using **tpsDig**, go to "Utility programs" and get **tpsUtil**, an enormously useful program that has many functions that make it easy to delete or reorder landmarks (or specimens) from your file of digitized data, as well as to prepare a file for data collection, append files, and to carry out other operations that will make your life much easier.

The next programs that you should get are the ones that will do the analyses. It may seem premature to get them before you have collected your data, but in the next section of this workbook we discuss how to load data files into the programs that do morphometric analyses so it is worth getting the programs now.

The tps series of programs can be found by following the link to "thin-plate spline" (which is what "tps" stands for). For purposes of this chapter, you can download any of them, but while you are there, you might find it most convenient to download the ones that you are likely to need in the future, including **tpsRelw** (which does a principal components analysis, and also does superimposition of semilandmarks, see Chapter 3), **tpsRegr** (which does simple regression and multivariate regression and multivariate analysis of (co)variance), **tpsPLS** (which does partial least squares) and, if relevant for your research, **tpsTree** (which reconstructs the evolution of shape on a cladogram input as a Nexus file). When you download and install these programs, they are all placed in the same folder, tps (unless you decide otherwise).

To get the other comprehensive software, follow the link to "Comprehensive software" and then to "MorphoJ" to get **MorphoJ** and "IMP" to get **IMP** programs. Clicking on these links will take you to the sites where you can download the programs. You could also get **MorphoJ** directly by going to http://www.flywings.org.uk/MorphoJ_page.htm . To run **MorphoJ**, you will need a version of the **Java Runtime Environment** (Version 5.0 or later). If you do not have it, you will need to get it first and install it before you download **MorphoJ**. You can download **MorphoJ** either by using the standard installer or the web installer (which will download the files in three small parcels, which is useful if your internet connection does not allow you to download 7MB files). Click on the desired installer and then follow the instructions. You can get the **IMP** software directly by going to http://www3.canisius.edu/~sheets/imp7.htm. Once at the **IMP** home page, follow the link to I**MP7** programs. These programs are written in Matlab, and due to Matlab licensing requirements, you will need to get a password from Dave Sheets to download **MCRInstaller**, which installs the components needed to run the compiled IMP programs. *Before downloading* **MCRInstaller**, create a folder for it (e.g., Matlab7). Once you have downloaded **MCRInstaller** into that folder, you run it and it creates a series of folders. You then put the program modules into the folder v78/bin/win32. The one program that you need immediately is **CoordGen7a**, but while you are there, also download **Regress7a** (which does regression), **PCAGen7a** (which does principal components analysis), **CVAGen7b**, which does canonical variates analysis and one-way multivariate analysis of variance) and **PLSMaker** (which does partial least squares analysis). Some of these programs can be copied from the downloaded folder, others are self-extracting files that are opened by clicking on the installer (make sure to install the programs in Matlab7/v78/bin/win32). If you want older **IMP** programs that are not yet recompiled for the most recent version of Matlab, they are on the "moremorph" page (the purple one); but you will need the older installer, **mglinstaller**, which is on the "morphsoft" page. You do *not* need a password to download this. The two versions of the software are downwardly and upwardly incompatible, so create a new folder, e.g., Matlab6, and put **mglinstaller** in that. Run it, then place the Matlab6 programs in Matlab6/bin/win32

To get **R**, go to http://cran.r-project.org/ (The Comprehensive R Archive Network). You want the precompiled binary distributions for your operating system. Click on your operating system and download the **base** package. Once you install it, you will want at least two of the contributed packages, **shapes** (not "shape," which is another package available there) and **vegan** (a package for ecological analyses that is useful (even necessary) for complex morphometric analyses). Open **R**, go to the toolbar and find **packages**; then go to **set Cran mirror** (pick the one nearest you), then **install packages**. It will give you a very long list of choices. Go to the first that you want and select it (it will download) and then the second. For purposes of practicing some basic operations in **R** (and for entertainment) also download the package **fortunes**.

Downloading a package will not make it immediately available to you. You will need to load it into **R** before you can use it. To load a package into **R**, at the command prompt (>) type:

```
library(fortunes)
```

or

```
require(fortunes)
```

Both functions will load a package (**require** checks to see if it is already loaded). At the command prompt, enter:

```
fortune(2)
```

You will see the text: “Bug, undocumented behaviour, feature? I don't know. It all seems to work in 1.6.0, so everyone should downgrade now... :)” followed by author and the quote’s source. To see another, use the up-arrow, which will bring up the last line that you typed. Replace the **2** by another number, such as **7**. Then you will see: “What we have is nice, but we need something very different.”

For a very reassuring perspective on research, try fortune(12).

It is unlikely that you would need to consult the manual for this function, but to see how to get help in **R**, at the command prompt, type:

```
?fortune
```

or

```
help(fortune).
```

If you need help on a function that isn’t in one of the packages that you have loaded (or if you can’t think of the name of a specific function and want a more general way to get help), enter

`??` (the name of the function or something that might find the function you want).

The other way to get help in **R**, aside from the manuals, introductions to **R** and mailing lists, is to do a web search on what you want to do, with “R” in the search, e.g.,

```
graphics in R
```

Loading data into programs: Files in tps format

The initial step in any analysis, and sometimes the most difficult one, is to load a datafile. This is difficult because many programs are finicky about file formats, and if the datafile is not properly formatted the program will not run. Some programs will give you an error message, others just won’t work. To practice loading a datafile into the programs that you downloaded, you need a datafile to load. We will use two, one that contains only landmarks (on the jaws of 10 prairie deer mice and 10 eastern fox squirrels) and one that also contains digitized semilandmarks. Both are “MouseSquirrel#.tps” available on the companion website well as at Dave Sheet’s website. The first one is *MouseSquirrel1.tps*, the second is *MouseSquirrel2.tps*. Both files are in tps format. To see what is meant by “tps” format, open *MouseSquirrel1.tps* in any text editor and look at the first specimen:

```
LM=15
966.00000 407.00000
1097.00000 566.00000
846.00000 496.00000
691.00000 541.00000
735.00000 626.00000
682.00000 632.00000
627.00000 639.00000
374.00000 863.00000
157.00000 892.00000
49.00000 884.00000
62.00000 527.00000
35.00000 842.00000
789.00000 616.00000
456.00000 457.00000
583.00000 638.00000
IMAGE=PM_5357_AR.jpg
ID=0
SCALE=0.002710
```

The file beings with the line LM=15, which means that there are 15 landmarks for this (and all other) specimens. Following that line are two columns of numbers. Each row has the *x*- and *y*-coordinates for one landmark; the first row is the first landmark, the second row is the second landmark, etc. Following the data for that specimen, you will see a line "IMAGE=PM_5357_AR.jpg", which is the label of the photograph that was digitized. That is followed by "ID=0" (because the program was written in a language that begins counting at zero). After that comes the scaling factor that converts pixels to absolute size (e.g., millimeters, SCALE=0.002710). The data for every specimen in this file begins with the control line "LM=15" and ends with the control lines "ID=#" (specimen id), "IMAGE=filename", "SCALE=#".

Now open *MouseSquirrel2.tps*. This is a much longer file. After you get past the first 15 rows of data, you will see a control line "Curves=6." This means that there are six curves on which semilandmarks were digitized. Right below that, you will see "Points=32." That means that there are 32 semilandmarks in the first curve. Below that are the 32 rows of coordinates for the semilandmarks on the first curve. Those are followed by "Points=32", which means that there are 32 points on the second curve. Because there are six curves with 32 points apiece, the "Points=32" line appears six times in the file. After the last coordinate for the last point in the last curve you will see the same final three control lines: "IMAGE=PM_5357_AR.jpg", "ID=0" and "SCALE=0.002710".

Every tps file begins with the control line "LM=#landmarks", and ends with the control line "ID=specimen#." (Those for SCALE= and IMAGE= are optional.) Between these, every tps file contains the two columns of *x*-, *y*-coordinates for the landmarks, with the first row containing the coordinates of the first landmark, the second row containing the coordinates of the second landmark. If files for 3D data are in tps format, there will be a third column for the *z*-coordinates. Virtually all tps files will also have the line "IMAGE=image filename" (the control lines do not need to be capitalized). There can be other control lines, including comments that you made about the specimens as you digitized (such as whether the photography is blurry, whether one landmark was difficult to see), other variables that you added to the file, etc.

What we described above is "tps format". Because **tpsDig** is the most widely used digitizing program, and the tps series was the first geometric morphometrics software, this file format has become the standard for 2D data sets. With one exception, all morphometrics programs that we know of can read this one (the exception is the **shapes** package in **R**). Another format produced by tps software is known as NTS or NTSys format. It too can be read by some morphometrics programs and if you save a file of centroid sizes from a tps program, that file will be in NTS format. In NTS format, MouseSquirrel1 looks like this:

```
1 20L 30 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
2.617860 1.102970 2.972870 1.533860 2.292660 1.344160 1.872610
1.466110 1.991850 1.696460 1.848220 1.712720 1.699170 1.731690
1.013540 2.338730 0.425470 2.417320 0.132790 2.395640 0.168020
1.428170 0.094850 2.281820 2.138190 1.669360 1.235760 1.238470
1.579930 1.728980 2.956024 1.207544 3.139068 1.710232 2.480656
1.371464 1.997092 1.344144 2.152816 1.636468 2.008020 1.633736
1.844100 1.603684 1.002644 2.079052 0.442584 1.999824 0.142064
1.887812 0.428924 1.013572 0.133868 1.764872 2.303076 1.631004
1.458888 1.073676 1.704768 1.606416 2.930790 0.958410 3.183588
1.397334 2.586318 1.150092 2.130726 1.194540 2.205732 1.477896
2.072388 1.475118 1.958490 1.472340 1.136202 2.105724 0.536154
2.072388 0.169458 2.002938 0.363918 1.058418 0.155568 1.936266
2.391858 1.458450 1.394556 0.952854 1.794588 1.477896
```

The first line contains the control lines that tell the program what kind of matrix you are analyzing, how many rows and columns it contains (and whether there are labels for either), and what option you are using for missing data. For the data files used in the tps programs, the first number is always a 1, the last one is nearly always a 0 (zero). **MorphoJ** can read a file that contains missing data, and if data are missing that last column contains a 1. The second number is the number of rows (i.e., specimens) and this is followed by an "L" if specimen labels are in the file. The third number is the number of columns (i.e., variables) and this is followed by an "L" if the variables are labeled. So the file we show here contains 20 individuals, labeled by their ID= line (which starts at 0), and contains 30 unlabelled variables, the *x*- and *y*-coordinates for 15 landmarks. It does not include centroid size. The paragraph breaks that you can see were not added by a text-editor, they were in the output file. They do not matter to any program that can properly read the control lines because the third number tells the program how many columns to read for each individual, and that many are read regardless of paragraph breaks.

To see how to load a tps file into a tps program, we can load *MouseSquirrel1.tps*, compute centroid size and save that to a file. Open the tps program **tpsRelw**. If the icon for it is not on your **Start** menu, go to **All Programs**, **tps**, and then **tpsRelw**. Open it. To load the data file, click the **File** button and navigate to the folder that has the datafile in it. Select the file. That loads it. This program does a principal components analysis of the data, but we will wait until Chapter 6 for that. For now, click the one enabled button, **Consensus**. When the next one is enabled, go to the **File** menu, go to the second from the last option, **Save** and select **Centroid size**. The file that you just saved is the file of centroid sizes, in NTS format. You should recognize that first line except that there is an (optional) title, set off by single or double quotes.

```
" Centroid size
1 20L 1 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

Loading tps datafile in other programs

Datafiles in tps format can be read by **CoordGen** (IMP), **MorphoJ** and **PAST**. These programs read files in other formats as well, which are described in the following section. For now, we will load tps files into each of them.

CoordGen

To load a tps file in **CoordGen**, first open **CoordGen** (go to your IMP7 folder, find **CoordGen7a.exe**, click on it and run it). You will see an orange box that lists a variety of file formats; if you used the scaling factor, you won't see that option in the Orange Box but you can find it in **File Options** on the toolbar. Go to that menu and select the first option **Load TPS file with Scale factor**. When you see an image of your data in the image window, your file is loaded. If you used a ruler instead of the scale factor (or if you used both and you want to find out if you can delete the ruler and save only the scale factor), you instead select the top option in the Orange Box, **Load TPS File (with ruler)**. Below the Orange Box you will see a Blue Box where you can tell **CoordGen** which two landmarks are the endpoints of the ruler and what the distance is between them. If you digitized different lengths for different specimens, you would need to put them into different files.

MorphoJ

To load a tps file into **MorphoJ**, first open **MorphoJ** (it should be on your **Start** menu or there should be a shortcut on your desktop, or both; if you don't find it, look at the **All Programs** menu and put the shortcut somewhere convenient). The first step is to create a new project (because you don't already have one); when creating it you will load this dataset and create the dataset as well. Go to the **File** menu on the toolbar, select **Create New Project**. A window will appear asking you to name the project and the data set, to specify the dimensionality of the data, and whether it has object symmetry, and the file format. To load *MouseSquirrel1.tps*, pick a name for the project that you can easily recall, e.g., MouseSquirrel. Use that as the name of the project and MouseSquirrel1 as the name for the data file. It is a two-dimensional dataset (the default), it does not have object symmetry,(also the default, we will talk about object symmetry in Chapter 3) and it is in tps format, so select that as the kind of file to load because it is not the default. Then navigate to the file and select it. In the **Report Window**, you will see "Reading from TPS file 'MouseSquirrel1.tps'. Finished reading the file." Now select the tab **Project Tree** and you will see the project name, and below that the datafile name. You will not see a picture of the coordinates until you request a Procrustes superimposition (covered in Chapter 3). If your data are 3D, say so when you load the file.

shapes(R)

Loading a tps (or IMP) file into **R** involves two steps. The first is to read the file into **R**, the second is to put it in the format read by Dryden's **shapes** package or Claude's functions (which use the **shapes** format). Loading a file in **R** is easy, but reorganizing the data into the shapes format is more challenging. There are three functions that load a tps file and reformat it, according to the details of the tps file (such as whether it has a scale factor, and/or curves and points). These are discussed in more detail in Chapter 3 when we use the **shapes** package and Claude's functions to superimpose landmark data. For the moment, we will first say how to load a file into **R**, in general, and then how to load a tps file and reformat it for the **shapes** package.

read.table is the function that we would usually use to load a standard datafile into **R**, by which we mean a file in which each individual's data is in a row, and the coordinates are sequenced *x*1, *y*1, *x*2, *y*2,…*xk*, *yk* (with the *z*-coordinates following the *y*-coordinates for each landmark in the case of 3D data). To load that file into **R**, at the command prompt you would enter:

```
data <- read.table(file.choose())
```

What you are doing is assigning the output of the **read.table()** function to a dataframe called **data**. The **<-** does the assigning. A dataframe can contain numeric data such as the coordinates of landmarks, and character data such as the names of your specimens, their species, their sex (m or f), etc. When you use **file.choose()** a window will open allowing you to navigate to your file and select it. It will then be read into **R**. You could, instead, write out the pathname of your file,

```
 data <-read.table (pathname)
```

or you could copy your file to the clipboard and paste its contents into **R** using

```
data <-read.table("clipboard")
```

We want to do more than just read the file into **R**; we also want to reformat it. The functions that we will use are not in any package that you download with **R**, rather they are written by users and made available to others. The functions that we will use to load a tps file into **R** and reformat it include one written by David Polly, **read.tps1()**, a version of **read.tps1()** modified by us to read the scale factor, **read.tps2()**, and a version modified by A. Michelle Lawing, **read.tps3()**, that reads the scale factor and the CURVES= and POINTS= lines, and converts the points along the curves into landmarks.

All three of these functions, plus the one that we will use to read an IMP file into **R**, plus many others, are in the folder **functionsR()**. Note that there is another folder, **scriptsR**. The scripts include more than user-supplied functions, they also contain commands and expressions to be interpreted line by line, plus comments (notes to yourself or anyone reading the script, which always begin with #). You will likely want to modify several lines in a script, because they can be datafile specific, and you may want to insert additional calculations or comments. Working with these scripts, or ones that you write yourself, and then saving the scripts, allows you to keep a complete record of your session.

You can open **functionsR()** either in **R** or in a text editor and look at it. If this is your first experience with **R**, you will likely wonder what to do with a function. The short answer is that you run them then apply them to your data. The first step is to read the function into **R**. You can copy the function and paste it into a new script window. To do this, copy the function that you want, e.g., **read.tps2()**, go to the **File** menu and select **New script**, and copy the text into that. Next, you want to select the lines to read into **R**; to select them all, right click and select the last option, **select all**, then right click again and select the first option, **Run line or selection**. To save that script, go to the **File** menu, select **Save script as** and give the script a name. The next time you want to use that script, go to the **File** menu and select **Open Script**, navigate to the folder where you put it, and select it.

If you have a large file of functions, and you want to make them all available for when you will need them, you can read them all into **R**, sourcing the code. To do that, go to the **File** menu and select **Source R code**. When the window opens, navigate to the folder where you put that file and select it. You will not see anything written to the console beyond the command prompt (>) and something like this:

```
("C:\\Users\\Mimi\\Desktop\\R_Stuff\\Morphometrics_functions.R")
```

However you read the function into **R**, you are now ready to use it to load your datafile and convert the tps file into a **shapes** file.

```
data <-read.tps2(file.choose())
```

To convert an IMP file, use the function **importToShapes()**, written by Adam Rountrey. As before, to use the function, first read it into **R** then run it.

```
data<-importToShapes(file.choose())
```

One thing to note is that the first letter is in lower case but the letters beginning the next two words are in upper case; **R** is case-sensitive so it matters whether you use upper or lower case.

PAST

To load your tps datafile into **PAST**, open **PAST** and then go to the **File** menu and **Open** your tps file. Save it as a **PAST** data file; on the **File** men, select **Save as** and name the file.

Loading datafiles in other formats

That so many programs can read a tps file is great if your datafile is in tps format. Although the variety of formats may seem overwhelming if you look at the several options on the **CoordGen** and **MorphoJ** input file menus, many differ only subtly and can easily be converted from one to another. A common feature of several file formats is that each specimen's data is in one row, with the *x*- coordinate for the first landmark coming first, followed by the *y*-coordinate for that landmark, then the *x*-coordinate for the next landmark and then the *y*-coordinate and so forth. IMP format differs from this only in that centroid size occupies the final column (after all the coordinates). The text-file format read by **MorphoJ** differs from it only in that specimen identifiers (e.g., museum numbers) occupy the first column (before all the coordinates). By deleting one column and adding another, simple operations that can be done in any spread sheet, these files are interconvertible. Some digitizing programs produce files in this format, but any format that can be read by either **CoordGen** or **MorphoJ** can be saved in this format (with an additional column for centroid size in **CoordGen** or specimen identifiers in **MorphoJ**).

CoordGen is the only IMP program that reads the tps file format, or any format other than IMP format. Thus, files produced by **tpsDig** (or other digitizing programs) would be loaded into **CoordGen** and the results saved to IMP format. Specimen identifiers can be added to IMP files, by adding a column of **%** after the last column, the one that contains centroid size. The **%** indicates that what follows is a comment (something to be read by you, not the program).

Three-dimensional data will rarely be in tps format. The datafile format for three-dimensional data appears to be less standard than for two-dimensional data, but **Landmark Editor** produces an “NTSys” output file that has the control line of an NTS file, followed by each landmark’s coordinates in a row (as in a tps file). In NTSys format, it does not matter whether each landmark’s coordinates are in one row, half a row or multiple rows because the control line includes a parameter that says how many columns of data there are for each individual; that many are read regardless of the number of lines they occupy.

Literature Cited

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

Dryden, I. (2012) shapes: Statistical shape analysis. R package. pp.

Hammer, Ø., Harper, D. A. T. & Ryan, P. D. 2001. PAST: Paleontological Statistics Software Package for Education and Data Analysis. *Palaeontologia Electronica* **4(1)**.

Klingenberg, C. P. 2011. MorphoJ: an integrated software package for geometric morphometrics. *Molecular Ecology Resources* **11**: 353-357.

R_Development_Core_Team (2012) R: A Language and Environment for Statistical Computing. pp. The_R_Foundation_for_Statistical_Computing, Vienna.

Rohlf, F. J. (2006) tpsPLS. pp. Department of Ecology and Evolution, State University of New York, Stony Brook.

Rohlf, F. J. (2010a) tpsDig. pp. Department of Ecology and Evolution, State University of New York, Stony Brook.

Rohlf, F. J. (2010b) tpsRelw. pp. Department of Ecology and Evolution, State University of New York, Stony Brook.

Rohlf, F. J. (2011) tpsRegr. pp. Department of Ecology and Evolution, State University of New York, Stony Brook.

Sheets, H. D. (2009) Simple3D7. pp. Canisius College, Buffalo.

Sheets, H. D. (2010) PLSMaker7. pp. Canisius College, Buffalo.

Sheets, H. D. (2011a) CoordGen7a. pp. Canisius College, Buffalo.

Sheets, H. D. (2011b) CVAGen7. pp. Canisius College, Buffalo.

Sheets, H. D. (2011c) PCAGen7. pp. Canisius College, Buffalo.

Sheets, H. D. (2011d) Regress7a. pp. Canisius College, Buffalo.

2

Obtaining Landmarks and Semilandmarks

This chapter is about two steps of data collection - the first is obtaining the images to be digitized, the second step is digitizing them. The **MOST** important decision that you will make about your data is whether to use two- or three-dimensional coordinates. This decision will affect both the information contained in your data and the mechanics of obtaining the data. There are many technologies for collecting 3D data, and this advances so rapidly that anything we say is likely to be obsolete shortly. For example, two years ago we might have said that microCT scanning was not well suited to very small specimens such as rodent teeth, and that the most useful technology was the Reflex microscope. But the development of high resolution computed tomography (sometimes called "nanoCT") now makes it possible to obtain 3D coordinates at the sub-micron level (Dhondt et al., 2010, van der Niet et al., 2010). As a result, structures of very small specimens including very fragile ones such as shoots and flowers of mouse-eared cress (*Arabidopsis thaliana*) can be recorded at a spatial resolution of 0.85 μm in a scan lasting approximately 1.5 hours. Of course, whether this method will be feasible for you depends on many factors aside from the size and fragility of your specimens - it also depends on whether you (or your specimens) can travel to a center where the technology is available (e.g. Belgium) and on your budget.

In general, the factors that determine the optimal technology for any research project include (but are not limited to): (1) specimen size relative to the resolution and accuracy of the device; (2) specimen fragility; (3) density of the tissues to image; (4) portability of the device and/or specimens; (5) ability to obtain images of specimens versus just coordinates of landmarks; (6) ability to image landmarks on internal structures versus just the external surface; (7) cost of buying the device or per specimen cost of having the imaging done; (8) labor; (9) training required to use properly the technology and/or software; and (10) time available for data collection. Two-dimensional data are far less expensive to collect, takes less time to obtain the images (and to digitize them) and the imaging device - the camera - is as portable as you are. At present, most studies rely on 2D data, especially when analyzing large samples of small organisms. Methods for collecting 2D data are now standardized whereas those for 3D data are not. We therefore only briefly summarize methods now available for 3D data collection and then concentrate on the most widely used method for acquiring 2D images (digital photography). The last part of this chapter is on digitizing coordinates of landmarks in the most widely used digitizing program, tpsDig (Rohlf, 2010).

Methods for obtaining three-dimensional data

Three-dimensional data can be obtained in two basic forms (but the details of both forms can vary considerably). The first is a set of coordinates digitized directly from the specimens - no images are recorded - only coordinates are. Technologies that produce only coordinates include the 3D digitizers, e.g. Microscribe (by Immersion Corporation), GoMeasure (http://gomeasure3d.com/wp/), and the Polhemus digitizer (by Polhemus), and the Reflex Microscope (by Reflex Measurement; http://www.reflexmeasurement.co.uk.). Below we call all of these the 3D digitizers because they are used to obtain the *x*-, *y*-, *z*-coordinates. The second form is a digital 3D image from which coordinates (and other information) can be extracted. Some imaging technologies penetrate the surface of the object and allow you to visualize both surfaces and internal structures. These methods include x-ray computed tomography (CT) and magnetic resonance imaging (MRI). Both produce “tomograms” - two dimensional cross-sections or “slices”; the 3D image is reconstructed from a stack of those slices. Below we call these the “penetrating scans”. Another imaging method, laser scanning, also reconstructs the 3D image from a stack of slices, but does not penetrate the surface. This we call “surface scanning”. A third kind of 3D imaging method works very differently from these others because it uses multiple 2D photographic images to reconstruct the 3D image (photogrammetry and computer stereovision).

Three-dimensional digitizing

One type of technology for 3D digitizing uses a stylus, which is connected to the base by a jointed arm (Microscribe; (http://www.emicroscribe.com/products/microscribe-mx.htm)) or a flexible cord (Polhemus; http://www.polhemus.com/), which allows you to reach around, behind and under the specimen. Digitizing is done by touching the tip of the stylus to the landmarks while the specimen is fixed in its position. The movement of the jointed arm is measured by its degrees of freedom (df); 6 df means that the arm can move forwards/backwards, up/down, left/right and rotate about three perpendicular axes, with the movement along the axes and rotation about the axes being independent of each other. 3D digitizers come in a variety of sizes and they vary considerably in accuracy; the most accurate one (Microscribe MX) promises to be within ±0.002 inches (in) or 0.0508 mm of the true value. The Polyhemus Patriot, promises less: an accuracy of only 0.05 in (1.3 mm). Even the most accurate 3D digitizer, however, may be unsuitable for small, fragile specimens because the specimen has to be fixed in its position - it must remain in precisely the same place while the stylus is moved around it. Fixing the specimen can damage fragile material and, as Hallgrimsson and colleagues (Hallgrímsson et al., 2008) point out, it can be a challenge to achieve the goal of a completely stationary specimen. For large specimens, these devices can be attractive because they are portable, weighing as little as 12 lbs (5.4 kg) in the case of the Microscribe MX, and they are also relatively inexpensive compared to other devices for 3D data collection. As priced today, the Microscribe MX, with 6 df, costs only $12,995.00 (http://www.emicroscribe.com/products/microscribe-mx.htm) and the Rhinoceros software adds only $995.00 to the cost. They are also relatively simple to use and the coordinates can be obtained quickly.

The other 3D digitizing technology is the Reflex Microscope, which is a stereomicroscope. When you look through the eye-pieces you will see a small dot, which you move into position over the landmark using a joystick that moves the stage on which the specimen rests in the *x* and *y* directions. To get the *z* coordinate, you move the microscope head up and down, using your stereoscopic vision to position the spot on the landmark, which can be especially challenging when the surface is oblique to your line of sight (if you have difficulty dissecting under a stereomicroscope, this technology may not be for you). When the dot is properly positioned (or rather, when the specimen is properly positioned), the coordinates are recorded by pressing a button on the joystick. The measurements have a very high degree of accuracy, down to ±3 microns in the *x* and *y* direction, and ± 5 microns in the *z* direction. Thus, very small specimens can be measured very accurately. The technology is not limited to very small objects, however. Even specimens as large as 110 mm x 96 mm x180 mm can be measured by this device. The size range that can be measured is described in the brochure provided by the manufacturer (Consultantnet Ltd) as from vole teeth to primate skulls. However, their primate is a rather small one, a monkey skull. This device is thus suitable for small to moderately large specimens. But even for small specimens, the device has some limitations. First, it is not portable. Second, it is fairly expensive; no price list is available on the manufacturer’s website, but one quoted by Dean (1996) in an overview of 3D technology, was $35,000.00 and a more recent quote on the Stony Brook morphometrics website, current as of 2001, is $41,000, although that includes a computer and software as well as the microscope and case. Although it is not possible to record a 3D image of the specimen, it is possible to record 2D images using the trinocular head that is available for video or photographic imaging.

Surface (laser) scanning

Various technologies are used by laser scanners, but the objective is to calculate *x*-, *y*-, *z*-, coordinates of an object by triangulation. A laser-emitter shines a light on the object, which is reflected onto a sensor (e.g. a camera lens). In the scanners used in morphometrics (as opposed to those used in surveying or hydrogeography and other long-distance scans), the distance between the laser and object is calculated from the position of the laser dot (or strip) within the camera's field of view. That position depends on the distance between the laser and the object and, because the distance and angle between the laser source and the camera lens are known, and the angle at which the light returns to the camera can be measured precisely, the distance between the laser-emitter and the object's surface can be calculated. In the long-range scanners, called "time of flight scanners", it is the time required for the reflected light to return to the sensor that is used to infer distance. The distance is calculated for a point, and by sweeping the laser beam across the surface, the distance can be calculated for a dense cloud of points. Ideally, scanning is done multiple times from many directions. The point cloud is then filtered, converted into a triangulated mesh and rendered into a 3D model by software such as Geomagic Studio (http://www.geomagic.com/en/products/studio/overview/), or Rhino (http://www.rhino3d.com/4.htm).

Scanners can be characterized by their resolution, accuracy, point density, depth of field and field of view. Resolution is the minimum separation between two points in the *x*-, *y*-plane that can be distinguished. This is affected by the width of the laser strip or dot, the distance to the object, the focal length of the lens and resolution of the charge-coupled device (the camera's sensor). Accuracy is the error in the *z*-direction, which is measured by the standard deviation of the difference between the measured distance to an object and the actual distance to it. Because accuracy is measured as an error, low values for accuracy (error) mean a more accurate estimate. It may be helpful to think of accuracy as a measure like resolution in that a lower number means higher quality data. The accuracy of a scanner is affected by optics, the system that does the calibration of the angles reaching the lens plus the laws of physics. The point density is the distance between neighboring points. Depth of field is the range of distances over which the scanner can obtain an accurate image. Outside that range, objects will appear blurry. The field of view determines the size of the object that can be imaged in a single scan, and this is usually expressed by an angle of a cone within which the object is placed for imaging. Large objects must be farther away from the scanner than smaller ones and accuracy decreases with that distance so scanners with large fields of view are needed for imaging large objects.

Laser scanners vary widely in the characteristics discussed above, and in price, portability and the size of the object that can be accurately scanned. They also vary considerably in the configuration of the device. One interesting model is the Kreon "Skiron" 3D laser scanner, which is maneuvered by a Microscribe 3D digitizer; instead of clicking on a landmark, the data collection has been compared to "spray painting" (http://www.b3-d.com/Products.html); the scanner collecting 45,000 points per second, at an accuracy of 16 μm (.0006 in). This is presently listed at $18,659.00 to $21,995.00 (the price does not include the Microscribe, which, for these systems ranges from approximately $6995.00 to $29,990.00). Laser scanners can be both far cheaper and far more expensive than this. Among the cheaper ones are some that are not limited by the size of an object because the object is placed in front of it rather than inside it. Among these are the NextEngine 3D Laser scanner http://www.nextengine.com/), costing only $4000.00 (with an accuracy of 125 μm (0.005 in). The eScan, which can be found for as little as $7795.00 (http://gomeasure3d.com/wp/products/lbp/3d-laser-scanners-pp/escan-3d-laser-scanners/) has a comparable accuracy of 150 μm (0.006 in), which can be improved to 89 μm (0.0035 in) with an advanced lens.

Three-dimensional penetrating scans: X-ray computed tomography (CT scans) and magnetic resonance imaging (MRI)

CT scanning and MRI are different modalities for obtaining 3D images of both surfaces and interior points. CT scanning works by measuring the attenuation of the x-ray by the object. The technique yields a set of slices ("tomographs') which are reconstructed into a 3D image. At one time, these were limited to slices along a single axis, but modern CT scanners use multiple detectors so the objects can be reconstructed in all directions. MRI uses different pulses of radiofrequency energy to map the relative abundance (and spin) of hydrogen nuclei in the presence of a strong magnetic field. The radiofrequencies realign the magnetization so that the nuclei produce a rotating magnetic field that is detected by the scanner. MRI is especially useful for distinguishing between soft tissues, but CT scans can also be used to visualize soft tissues, and MRIs can also be used to visualize bones and teeth (and both can be used on fossils). The stacks of slices produced by both methods are reconstructed into 3D images by software such as Amira (http://www.amira.com).

CT scanners are more widely available, faster and less expensive than MRIs. MRIs are preferred in medical applications when the specimens (patients) are living animals because MRIs do not expose them to repeated doses of ionizing radiation. Both techniques have two major advantages over the other methods discussed above. First, unlike the 3D digitizers, they produce images as well as coordinate data and, unlike the surface (laser) scanner above, they provide information about internal as well as external anatomy. Both have the disadvantage of being extremely expensive and of not being portable although you may not have to take your specimens very far if your institution has a medical or dental school that allows researchers to analyze (non-human) material. Usually, you would pay for the time of the technicians who do the scanning plus for the software to reconstruct the images (and for a technician or research assistant who is trained in using that software). Often, the time needed for scanning can be reduced by packing many (small) specimens into a single tube.

Photogrammetry and stereovision

Photogrammetry and stereovision are techniques that rely on photographs to reconstruct three-dimensional shapes. Like laser scanning, these techniques reconstruct only surfaces. Also like laser scanning, triangulation is used to estimate the coordinates of landmarks (taken from multiple views). Photogrammetry emcompasses a range of techniques that estimate the three-dimensional coordinates from landmarks that are common to multiple photographs. The base of the triangle is the line between the camera positions, and the sides of the triangle are rays that extend between each camera position and the landmark on a specimen. This technique thus requires identifying landmarks that are visible in each view. The coordinates are estimated as the points where the rays from camera positions to landmark intersect. The estimates are affected by parameters intrinsic to the camera, especially its focal length and lens correction as well as extrinsic parameters like the position and orientation of the camera. Both intrinsic and extrinsic parameters must be calibrated before the coordinates of the landmarks can be estimated, and the intrinsic parameters of the camera must be held constant between the calibration step and data collection and throughout data collection (hence zoom lenses are discouraged). After photographing a known object multiple times in multiple views, the intrinsic camera parameters can be determined. Once both intrinsic and extrinsic camera parameters are estimated, a 3D surface consisting of polygons connecting the landmarks is reconstructed.

A surface reconstructed from just a few polygons will not accurately reconstruct a complex, curving surface just as a complex 2D curve will not be well reconstructed by a small number of straight lines. A landmark-poor morphology is not a good candidate for this method. It is, however, an inexpensive method for obtaining 3D data and the system is highly portable. It can even be used under field conditions - one of the most interesting applications is a field study of tortoise carapace morphology (Chiari et al., 2008). The live animals were photographed under natural conditions; as the authors point out, these animals do not move quickly. However, their carapaces offer few landmarks; the landmarks were located at intersections between scutes and only two landmarks were at intersections with peripheral scutes. As a result, the deviations between geodesic distances measured by flexible tape and those reconstructed by photogrammetry were exceptionally large, on the order of 15-20 mm.

Stereovision differs from photogrammetry primarily in the density of points used in reconstructing the 3D coordinates. Like photogrammetry, it uses partially overlapping views of an object, but these views are stereoimage pairs captured by two cameras at a known distance from each other (they may even be bolted to each other). Multiple partially overlapping stereoimage pairs are taken, and these are used to reconstruct the portion of the specimen that is visible in both images. During the processing stage, the reconstructed partial images are aligned with each other (by specifying the locations of three corresponding points in the pairs). The method is therefore not limited by the number of landmarks or their spacing because landmarks are necessary only for aligning the partial models and only three (per partial model) are used for that purpose. This method may be less useful under field conditions, but Chiari and colleagues found that the method performs better than photogrammetry.

Both approaches are relatively inexpensive. The only costs are the camera(s) and modeling software. For stereovision modeling, two cameras are needed as well as the software, but the pair of cameras (synchronized stereo head, with a variable baseline) made by Videre Designs are listed at $2000 (http://users.rcn.com/mclaughl.dnai/) and bundled with SRI International's Small Vision System. Photogrammetry requires only one camera; the program widely used for reconstructing 3D images, Photomodeler, is presently listed at $1145.00 (http://www.photomodeler.com/index.htm).

Obtaining two-dimensional data

The important (and obvious) difference between 2- and 3D data collection is that 2D photographs of 3D objects project that object onto a plane. If objects are not consistently oriented, the projected images will seem to differ in shape even when they differ solely in orientation. Thus, the first step in 2D data collection is to figure out how you will ensure a consistent orientation for your specimens. That will obviously depend heavily on the morphology of your specimens. If they are truly flat, or nearly so (such as the mandibles of mice split at the symphysis), you can just place them on the copy stand alongside a ruler (or even use a flat-bed scanner). However, if they are more nearly round, such as the crania of mice, you will need both a method for positioning them and for maintaining them in place. If you are traveling to various museums to collect data, you need a portable device that can ensure consistent position. Before you begin to collect data, you should take multiple photographs of the same specimen, placing it into position for each photograph, so that you can estimate the error due to positioning. If you photograph the same image 10 times, and digitize each image 10 times, you can assess the error due to positioning and digitizing (by the variation explained by each).

Included below are the rudiments of taking a picture (what makes the image and how can you manipulate the camera and the lighting to get a better image), and a few things you can do to make the captured image even better in photo editing software. Although the range of hardware and software has become fairly standardized, most of the following discussion is quite general. You will need to familiarize yourself with the characteristics of your particular system (but then, you will need to do that anyway to get the best possible results). Our goal is to provide you with an orientation to the subject that gets you through the first stages of that familiarization with a minimum of unnecessary pain.

Inside the camera

The aperture. Arguably, the most important part of a camera is the aperture - the small hole that lets light into the box to form the image. The aperture is critical because light reflecting off the object is leaving it in many different directions. The aperture functions as a filter that selects light rays based on their direction of travel. The only rays admitted into the camera are the ones traveling on the path that takes them through the aperture. In theory, if the aperture is small enough (and nothing else intervenes), it insures that the geometric arrangement of the rays' starting locations is exactly reproduced by the geometric arrangement of the rays' arrival locations (Figure W2.1). This is why a child's pin-hole shoebox camera works. It is also why the image is inverted.

Because the image is formed from a cone of light leaving a three-dimensional object and arriving on a 2D surface, there are certain artifacts or distortions introduced in the image (Figure W2.2). As discussed below, a good lens system can reduce these effects, but it cannot eliminate them completely. One distortion in photographic images of 3D objects is that an object closer to the camera will appear to be magnified relative to an object that is farther from the camera (Figure W2.2A). This occurs because the light rays traveling toward the lens from opposite corners of the object form a larger angle when the object is closer to the aperture, which means they will form a larger image inside the camera. For this same reason, the closer feature will hide more distant features. Another distortion is that smaller objects in the field may appear to be behind taller objects when the smaller objects are actually next to the taller object (Figure W2.2B). The reality is that the smaller object is farther from the aperture, but not in the expected direction. In a related phenomenon, surfaces of an object that face toward the center of the field of view will be visible in the image, and surfaces that face away from the center will not be visible

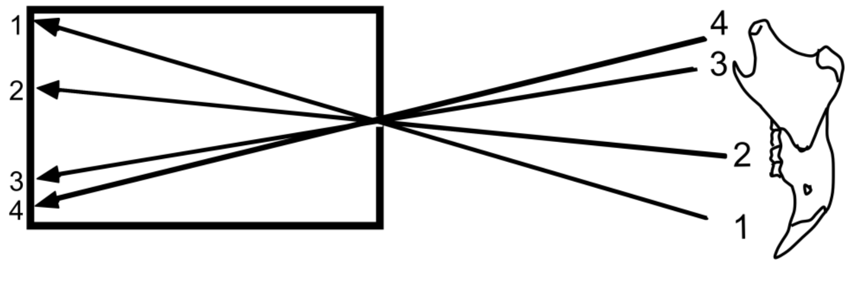


Figure W2.1. Image formation in an idealized pin-hole camera. Light rays travel in a straight line from a point on the object (the squirrel jaw) through the aperture to a point on the back wall of the box. The geometric arrangement of the starting locations of light rays and their reproduction by the rays' arrival locations.

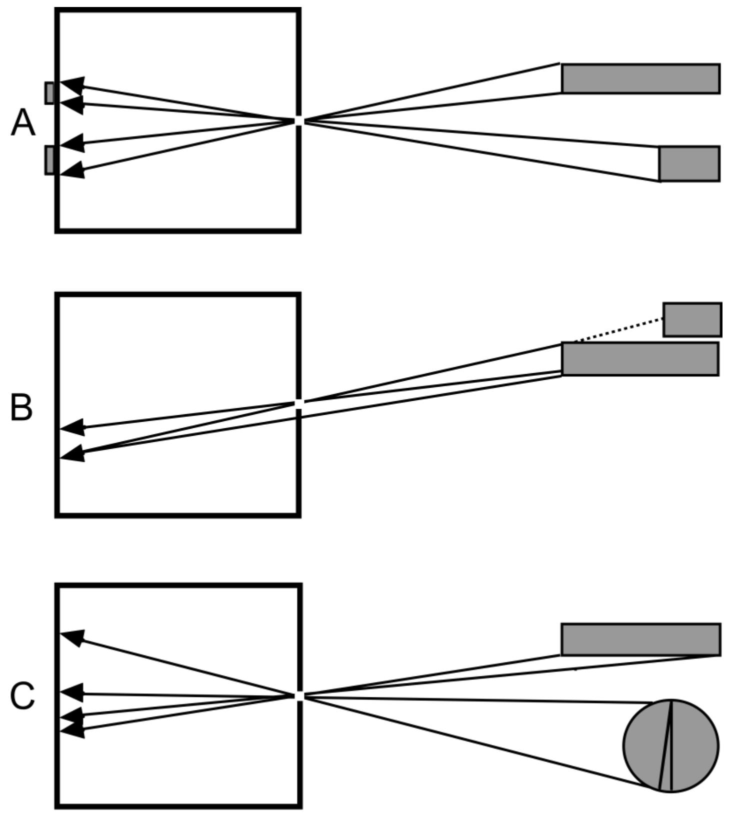


Figure W2.2. Distortions resulting from light leaving a three-dimensional surface and arriving on a two-dimensional plane. A: The two rectangles have the same width but the upper ("taller") rectangle produces a larger image (appears to be magnified) because its end is closer to the camera. B: An object that is farther from the center appears to be behind an object that is closer to the center of the image, especially if the more central object is "taller". C: The sides of an object that face the center of the field are visible and the surfaces that face away from the center are hidden. In the special case of a spherical object, less than half of the surface will be visible; if the object is not centered in the field, the apparent horizon will be tilted away from the expected reference plane (the equator) toward the aperture.

(Figure W2.2C). This is the reason buildings appear to lean away from the camera in aerial photographs. In the case of a sphere, this means that the visible edge (horizon) will not be the equator, but will be closer to the camera than the equator. If the sphere is not centered in the image, the horizon will also be tilted toward the center of the image (this effect can be a serious obstacle to digitizing landmarks on the sagittal plane of a skull). These phenomena are more pronounced near the edges of the image, so one way to reduce their influence on your results is consistently to center your specimens in the field of view. Near the end of the next section we discuss other steps you can take to minimize these distortions and the effects they would have on your morphometric data.

What the lens does. The lens does two things: it magnifies the image, and it makes it possible to use a larger aperture than a pin-hole. Both are important advantages, but they come with a cost. The size of the image in a pin-hole camera is a function of the ratio of two distances: (1) the distance from the object to the aperture; and (2) the distance from the aperture to the back of the box. If the object is far from the pin-hole, light rays converging on the aperture from different ends of the object will form a small angle. The light rays will leave the pin-hole in the same small angle, so the image will be smaller than the object unless the box is very large. One way to enlarge the image is enlarge the box, another is to shorten the distance between the camera and the object, so that light rays converging on the aperture from different ends of the object will form a very large angle covering of the back surface (Figure W2.3A). A lens magnifies the image by changing the paths of the light arriving at the lens so that the angle between them when they depart the lens is greater than the angle between them when they arrived. Consequently, the image is larger than it would be without a lens, making the object appear to be closer to the lens than it is (Figure W2.3B).

The amount of magnification produced by a lens depends on several factors. Light striking the surface of the lens at 90° does not change direction but, as the angle of incidence becomes more acute, the change in direction increases. Exactly how much the path of the light is bent depends on the properties of the material of which the lens is made and on the wavelength of the light. The light changes direction

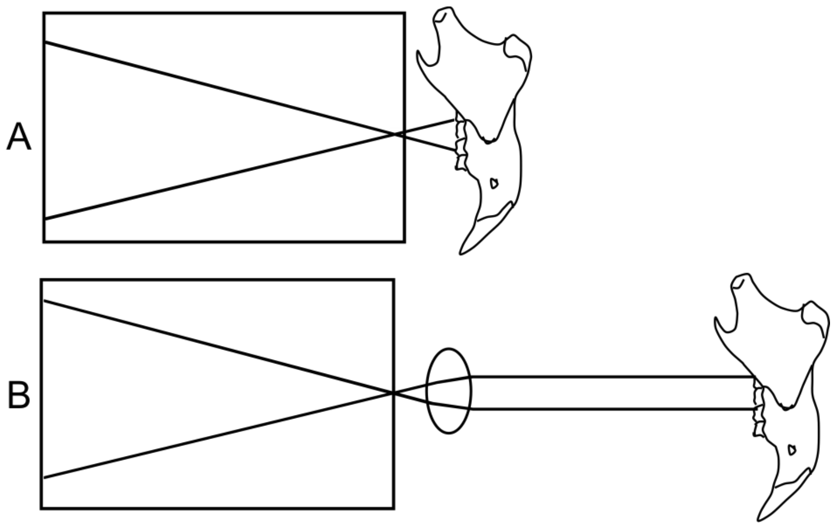


Figure W2.3. Two methods of image magnification. A: Moving the camera and object closer together. B: Using a lens to change the paths that the light travels from the object to the image, thereby changing the apparent distance of the object from the aperture.

again when it exits the lens. In addition to the advantage of having a larger image, which enhances resolution, there is the additional advantage that the distortions that occur in images of 3D objects are reduced. Because the object is farther away than it would be for a pin-hole image of the same magnification, the same small aperture is now selecting a narrower cone of rays leaving the object. This is particularly true for features near the center of the image. Features near the edges of the image are subject to other distortions (see below).

The image in a pin-hole camera is faint because the pin-hole must be small to be an effective filter of the light rays' directions of travel. A larger aperture would admit more of the light leaving the object, but it would produce a fuzzier image because a larger cone of light leaving each point on theobject would reach the back of the box (Figure W2.4A). Consequently, features in the image would have wide diffuse edges, and the edges of adjacent features would overlap, making it impossible to discriminate between those features. The lens corrects this problem by bending the light so that the cone converges again at some point on the other side of the lens (Figure W2.4B). This allows you to increase the size of the aperture, allowing more light from the object to reach the back of the box. The image that results is generally brighter and has more contrast between light and dark areas.

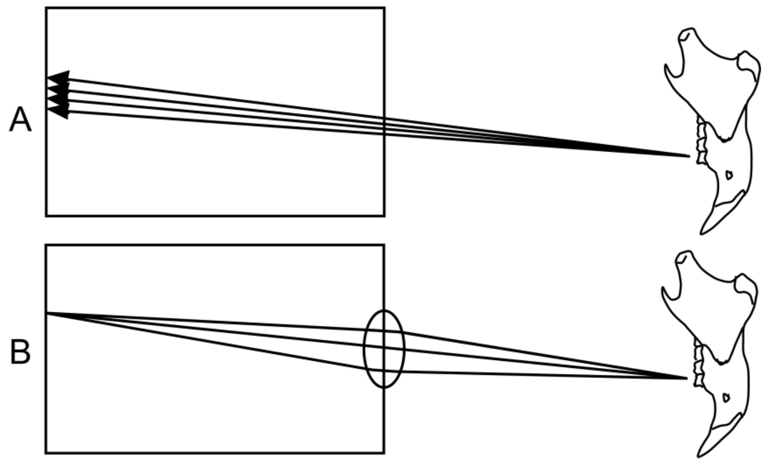


Figure W2.4. The role of the lens in enhancing image resolution. A: A large aperture admits many rays leaving the object in divergent directions, which produces a fuzzy image because each point on the object produces a relatively large circle of light at the back of the box. B: The lens bends the light so the diverging rays from a point on the object converge on a point at the back of the box.

The principal cost of using a lens is that it imposes a particular relationship on the distances from the lens to the object and the image. This relationship is expressed by the following equation:

$$1/f = 1/do + 1/di \qquad (2.1)$$

Where do and di are the distances to the object and image, respectively. The value of f is determined by the shape and material properties of the lenses, and is called the focal length. For the cone of light from a particular point on the object to converge again at the back of the camera, that point on the object must be a specific distance from the lens. At this distance, that point is “in focus.” If a part of the object is not at this optimal distance, light leaving that part does not converge at the right distance behind the lens, and that part of the image is blurred. The thickness of the zone in which this effect is negligible is the depth of field. Greater depth of field means that a thicker section of the specimen will be perceived as in focus. Depth of field decreases with magnification. At higher magnification, the light is bent more as it passes through the lens, so the difference in focal points is magnified as much as the areas of the features. Consequently, a thinner section of the specimen is in focus. To complicate matters further, in simple (single-lens) optical systems, the slice that is in focus is curved, not flat. Similarly, the surface on which the image is in focus is also curved. The complex lens systems of higher quality optical equipment flatten these surfaces considerably, but you may still find that only the center of a flat object (or a ring around the center) is in focus. The best solution for this problem is to use a lower magnification, increasing the depth of field. There are things you can do to edit the “captured” image but, as we discuss in a later section, these are limited by the initial quality of the image.

Near the edge of the image, additional distortions produced by the lens may become apparent. These distortions arise because the amount that the path of light is bent is not just a function of the properties of the lens. The deflection is also a function of the angle of incidence and the wavelength of the light. Two rays arriving at the center of the lens from locations near the center of the field of view are bent by relatively small amounts because they strike the surface at nearly 90°. Two rays arriving at the lens from locations near the edge of the field of view are not only bent by larger amounts, the difference in how much they are bent is also greater. Consequently, a straight line passing through the field will be curved in the image unless it passes through the center of the field. Closer to the edge of the field, differences in how much different wavelengths of light are bent by the lens may also become apparent as rainbows fringing the edges of features in the image. This is effect is most evident under high magnification or very bright light.

Checking your system

The complex lens systems of higher quality optical equipment greatly reduce all of the distortions discussed above, but none of these distortions can be eliminated completely. Fortunately, there are a few simple things you can do to insure that the effects on your data are negligible. The first is to put a piece of graph paper in the field and note where the rainbow effect, if any, becomes apparent. Next, digitize several points at regular intervals along a line through the center and compute the distances between the points. As you approach the edge of the image, the interval will gradually change. Take note of where this effect begins to be appreciable; you will want to keep the image of your specimen inside of this region. In other words, if the object is large, place the camera at a greater distance so that the image does not extend into the distorted region of the field. Next, get a box or other object with a flat bottom and vertical sides and mark one side of the box at a height corresponding to the thickness of your specimens. Put the box in the field of view, with the marked side at the center of the field. Slowly slide the box toward one edge of the field until you can see the inner surface of the side between the mark and the bottom. Again, you will want to keep your specimens inside this region. Outside of this region, features at the height of the mark will appear to be displaced away from the center of the image. Finally, check your depth of field by putting a sloped object marked with the thickness of your specimen (or one of your larger specimens) in the field. If all the critical features are not in focus at the same time, you should use a lower magnification to avoid guessing where in the fuzz is the feature you want to digitize.

What happens in the back of a camera

Now that you have a minimally distorted image at the back of your camera, you need to "capture" the image with a light sensitive device (the detector array). These contain a large number of light detectors used to record the image (pixels); a bundle of three light-sensitive devices recording intensities in three narrow color ranges. The higher the number of detectors, the higher the resolution of the recorded image. The image captured by your camera is not the image you digitize. What you see on the screen is a second image reproduced from the information collected by the detectors. If the camera image is mapped to the screen 1-to-1, pixel for pixel, the two images will have the same resolution. You can enlarge or reduce the image but you cannot increase its resolution. When the screen image size is reduced, information from multiple camera pixels is averaged for display by a single screen pixel. This may produce an image that looks sharper, but that loses the small-scale, almost imperceptible details that created the original "fuzz". The reduced picture may be easier to interpret but at the risk of merging features you want to digitize. When the screen image is enlarged, information from a single camera pixel is displayed by multiple screen pixels. This produces the blocky, stair-step effect. The result seems less resolved because the edges are not smooth, but features that were separate before still are separate. The drawback is that excessive enlargement may make the image difficult to interpret and increase the mental strain of digitizing.

Saving image files

Once you have an image "captured", you must decide how you want to save it. One option, often the default, is to use your camera-specific format for raw (unprocessed) photographs, e.g. *.NEF for Nikon, *.MRW for Minolta). In general, these contain more information than any processed photograph;

they are 16-bit rather than 8-bit, and therefore have more detail. Most raw files can be read by photo-editing programs, although which ones can be read depends on the photo-editing programs (and the freeware programs may be more limited than the ones that you pay for - so check your photo-editing program if you do not want to invest in another). As of this writing, the raw formats cannot be read by the tpsDig. If you save your images in the camera-specific raw format, you will later need to convert it into a standard format (e.g. *.JPG, *.TIF) but you can take advantage of the greater information contained in the raw photographs when enhancing the image. However, if you cannot easily download the images when the camera is full, you might prefer to save smaller files, e.g. *.JPGs, while taking the photographs. You will lose information when the file is compressed into a JPG, and you will actually lose information each time you save a JPG because it is compressed each time it is saved. You might instead want to save TIF files (but these too are very large).

Most standard image file formats are raster formats (also called bitmap formats). In these formats, the image is represented as a set of values assigned to a grid. This format reflects the structure of your screen and the detector array in your camera. BMP, TIF and JPG are all raster formats. The principal alternative is the vector format in which the image is represented by a series of mathematical formulae that specify a set of geometric shapes. This format has some advantages over the raster format, but it does not work well with photographic images of biological specimens because the complexity of these specimens requires a large number of geometric shapes to be mapped onto the specimen. Meta formats, such as that used in Windows metafiles (*.WMF), allow data in multiple formats in the same file, permitting the user to build up complex compositions (e.g. a picture, plus a graph, plus text).

The quality of an image reproduced from a raster file depends on the number of bits used to save the information at each cell (pixel) in the grid. The number of bits determines the number of colors or gray tones in the image. A 16-bit image can contain up to 64K colors, an 8-bit image can contain only 256 colors. Each pixel displays only a single color, so the advantage of the 16-bit image is that it can have much smaller changes in color from one pixel to the next. Thus, the 16-bit image can more accurately reflect graded changes in color across the object. In practice, the 8-bit image may not be noticeably poorer unless the image size is changed, and the 8-bit image file would have the advantage of requiring much less disk space in your camera (and on your hard disk). The most economical format is JPG (Joint Photographic Experts Group, JPG). This is a compressed format analogous to the *.ZIP format. An image that requires 900K of disk space as a 16-bit BMP or TIF file might require less than 100K as a minimally compressed JPG. More important, the 100K JPG will look just as good on the screen because there is very little information lost in the compression. In contrast, a 4-bit BMP file requiring about the same disk space will have lost much more information and look considerably worse.

If you have room in your camera's memory, keep the files in raw format and save them in that format to your hard disk. Then you can convert the files to JPGs without losing the original raw file.

If you are really pressed for disk space but need to preserve as much color information as you can, explore the options in your software. Normal color reduction replaces each pixel with the nearest color in the reduced color set (e.g. emerald, jade and lime will all be replaced with green). This creates large blocks of uniform color that obliterate many details. Various optimizations and diffusion algorithms produce "speckled" images that blend into more natural colors when viewed at a distance, or when reduced. These also do a better job of preserving edges.

Improving the image

Before you take the photograph

What you see in the image depends on how much light you shine on the object and how much of the light reflected from the object you allow to reach the detector. There are several options for manipulating light; the trick is to find the right balance so that you can see the features you want to digitize. It is important to understand that the best image for digitizing may not be the most esthetically pleasing image.

When you shine a light from a single source on a three-dimensional object some parts are likely to be in shadow. Shadows can be advantageous in that they allow you to see the relief; but you want to avoid a shadow so dark that you cannot see anything *in* the shadow. Backlighting allows you to see features in the shadow without obliterating the shadows. This is achieved by using a weaker light, or some kind of reflector (e.g. a piece of white paper) to illuminate the "back" of the object.

The size of the aperture and the amount of time it is open determine the amount of light that strikes the detector. A larger aperture admits more light but, as discussed above, the image is less sharply resolved. However, minimizing the aperture does not necessarily produce the most useful picture. A small aperture allows very little light to reach the detector from any area and the resulting image is generally dark. You can compensate for this by decreasing the shutter speed (or its digital analog). This allows light through the aperture to register on the detector for a longer period of time. As the shutter stays open longer, the brighter parts of the specimens become brighter in the image and the dark parts of the image stay dark. In other words, the contrast is increased. Unfortunately, minimizing shutter speed does not necessarily produce the best picture either. The longer the time that light is collected, the longer the fuzzy fringes register on the detector. If you leave the aperture open too long, eventually thin dark features will be obliterated completely and small bright areas will appear larger than they really are. You can also compensate for small aperture size by using brighter lights to illuminate the object. This has much the same effect as increasing the time the aperture is open. More light registers because there is more light from the specimen per unit time.

In summary, getting a decent picture may require a delicate and sometimes annoying balancing act. We strongly recommend that you try many different settings to see what works best and keep a log of the conditions in which each picture was taken. In your log, you should also take note of the brightness and shininess of the specimen. A dull gray specimen may require a different set-up than a shiny white specimen. You should also take note of what other room lights are on. If the room where you are working has windows, the time of day can be an important factor, as well. Have patience. Although there is a lot you can do to edit an image, you can only highlight information that is already there. You can’t recover information that was lost by the original.

After you have taken the photograph (photo-editing).

A quick tour through almost any photo-editing software will reveal a bewildering array of functions you could use to modify your image. Here, we discuss a few tools that are widely available and will likely to be useful to a large number of biologists. Many of these manipulations reduce the accuracy of the image as a reproduction of the original image. Again, it is important to realize that an esthetically pleasing or artistically interesting image may not be the best one to digitize for a morphometric analysis.

Probably the two most generally useful tools are the ones that adjust brightness and contrast. These functions can be most easily understood if your image editor displays a histogram of pixel luminance (the intensity of light emitted). Increasing brightness makes the whole image lighter, adding the same increment of luminance to every cell, up to the maximum value. Detail is lost at the bright end because pixels near that end converge on the maximum value. Details at the dark end may emerge as they are brought into a range where the differences between adjacent cells become perceptible. Except for the pixels near the bright end, the actual difference in brightness between adjacent cells does not change (the peaks in the histogram move toward the bright end, but they do not change shape). Decreasing brightness has the opposite effect. Increasing contrast makes the dark areas darker and the bright areas brighter, shifting the peaks away from the middle, towards the ends. Decreasing contrast makes everything a homogeneous gray, shifting the peaks toward the middle. The peaks also change shape as they move, becoming narrower and taller with decreasing contrast, and wider and flatter with increasing contrast. Again, differences between adjacent cells are lost as their values converge on the ends or the middle. Adjustments of either brightness or contrast can be used to make features near the middle of the brightness range easier to distinguish. The difference is whether the features that are made harder to distinguish are at one end (brightness) or both ends (contrast) of the range.

As noted above, raster formatted images have jagged edges. When the image is scaled up, it is also apparent that sharp edges in the original are represented as transition zones with large steps in brightness and/or color. This creates the problem of deciding exactly where in the zone is the edge you want to digitize. Adjusting brightness is unlikely to solve this problem because the number of steps and the difference between them stays the same. Increasing contrast can help more, because it makes the steps bigger. This comes at the expense of making the jaggedness more apparent. Even so, narrowing the zone of transition may be worth the increased jaggedness. Some alternatives to increasing contrast may include sharpening and edge enhancement. These tools use more complex operations that both shift and change the shapes of the luminance peaks, but they also can produce images with thinner edges. In general terms, the effect is similar to increasing contrast, but the computations are performed on a more local scale. Which tool works best to highlight the features you want to digitize will depend on the composition of your picture. Consequently, what works for one image may not work for another.

Additional photo-editing steps

As well as enhancing the images, you may want to edit the photographs further to make digitizing easier. One step that does not alter the image - renaming the files so that each one has a unique (and, if desired, recognizable) name. The others do alter the images. One step that makes the files easier to load and also to navigate is cropping; this eliminates the excess background, which both reduces the size of the file and makes it unnecessary to hunt for the specimen as you scroll through the file (which you will do when digitizing). Another is flipping or otherwise reorienting the photographs to standardize the orientation that you see when you digitize the image, which also standardizes the physical process of digitizing. Finally, you can resize the images if the files are very large by fixing the number of pixels (either by width or height).

Renaming the photographs with a recognizable and unique identifier, such as the specimen name or number, serves two purposes. First, it solves the problem of having several specimens with the same filename (e.g. DSC_0001.JPG). Each time you download the photographs from the camera, it will restart at DSC_001.JPG and if you save the new DSC_001.JPG to the same folder, it will overwrite the one already there. You can avoid that problem by keeping the images in different files, but that will eventually create a problem when you digitize the images - either you will need to digitize several folders of files separately or you will have to include the full path name of the file in the file name. Second, naming the files with relevant information allows you to recognize the images as you digitize them and to sort data files later. The filename is shown along with the photograph when you are digitizing them in the most commonly used digitizing program **tpsDig** (see below), and it is included in the output of that program. Additionally, you can get a list of all the filenames of the photographs using another program in the tps series, **tpsUtil**. If the name of the image file contains useful information, you will know what you are digitizing while you are digitizing, and you will able be able to sort your data file as you add (or remove) specimens from them. If you do not want to know what you are digitizing while you are digitizing, you can give each specimen a code instead of a recognizable label.

Cropping is useful if your specimens take up relatively little space in the picture or, more importantly, if they take up different amounts of space. Cropping them so that all specimens take up approximately the same space in the picture, and resizing the images to a standard image size, makes it easier to go from one picture to the next without adjusting the magnification in tpsDig. It is particularly useful to crop and resize when image files that are larger than about 4M open slowly (if at all) in **tpsDig**. However, if you did not include a ruler in the image, but did photograph them all at a standard distance (and magnification), do not adjust the size of the photograph. Under those conditions, the only information that you will have about size is the actual size of the photographed specimen. You should crop the images before enhancing brightness and contrast so that you are enhancing the image of the specimen not the background. Once cropped, it is useful to resize the images; saving the processed files as TIFF files will ensure that they will not lose information. But TIFF files are very large. After cropping the images, you can save the image at 1000 pixels wide.

Reflecting (or perhaps also rotating) the specimens is useful if they were photographed in different orientations. You do not need to perform either step. You can reflect specimens (i.e. flip them either horizontally or vertically) while digitizing. In **tpsDig**, there is a function in the **Image tools** menu (on the **Options** menu) for flipping the images, which are then saved by going to the **File** menu and selecting “**Save Image**”). A particularly useful feature of this program is that you can flip the images *after* digitizing them because the data will flip with the specimen. You can flip them later after you have finished digitizing the specimens by opening the file in a spreadsheet and multiplying the *x* or *y* coordinates of every landmark by -1 according to the orientation that you want to reverse. However, if half the specimens face left and half right or half face up and the other half down, digitizing is less straightforward because you will not develop a reliable search image and the physical process of digitizing (especially that of tracing curves) can introduce subtle variation in the handedness of the images that might look like asymmetry of the specimen. This is why you might also want to standardize the orientation of your specimens if you did not do that while photographing them. If you are digitizing landmarks that are recognizable as inflection points on curves, having the curves in different orientations may make it difficult to recognize those inflection points.

After editing the images, rename them and save them in the desired format; TIF files have the advantage that they lose no information when you reopen and close them, but they have the disadvantage of being very large. If you plan to save them as TIF files, resize the image (e.g. to 1000 pixels wide). If your program offers you the option of using LZW compression, do not select it; the files will not open in **tpsDig**.

There are many photo-editing programs, just make sure yours does not lose information, or worse, distort the image, which can happen if you rotate them. Microsoft viewer is not adequate for this purpose, but you can find freeware photo-editors that are (e.g. GIMP, http://www.gimp.org/) Paint.net (http://www.getpaint.net/) or Serif PhotoPlus Starter Edition (http://www.serif.com/FreeDownloads/). An important limitation of free programs is that they usually do not allow you to write scripts for repetitive tasks (e.g. adjusting brightness and contrast, rotating the images, sizing them to 1000 pixels, and saving as TIF files). In some cases, features (including scripting) are disabled to encourage you to pay for the full version. The full version of Serif PhotoPlus is presently $89.99, which is modest compared to PhotoShop (presently listed at $699.00 http://www.adobe.com/products/photoshop/buying-guide.html). If you find yourself spending a lot of time editing your photos, you should look at several of the freeware programs because some are more intuitive than others. Look for a program that allows you quickly to crop, adjust levels and contrast, flip the image both vertically and horizontally, resize the image and save it to your desired file type. If you save your images in raw format, you also want to make sure that the program can read it.

Digitizing in tpsDig

This discussion of digitizing focuses on how to use one particular program, **tpsDig.** We recommend it not only because it is an excellent program but also because virtually all programs for shape analysis can read data in the format output by this program. Therefore, you will not need to reformat the data before you can analyze it. Also, t**psDig**, especially the most recent version of it, includes useful functions that other digitizing programs do not have. As of this writing, there are two versions of this program available on the Stony Brook morphometrics website, **tpsDig1.4** and **tpsDig2.16**. In general, **tpsDig2.16** is the more useful one, but it is also worth having **TpsDig1.4** because you may find that one essential task is easier to perform in that version (set scale), another is more convenient to use in that version (the template function), one editing step is easier to do in that program (insert landmark) and one other is often possible only in that version (delete landmark) after the semilandmarks have been resampled.

We first describe how to prepare the data file before digitizing, using another program in the tps series, **tpsUtil**, an exceedingly useful utility program. Once that empty data file is made, it is then opened in **tpsDig** and the coordinates are written to that file. Finally, we describe how to digitize and edit the semilandmark curves and how to turn them into data (they are simply background curves when you first digitize them). Semilandmarks do not have to be digitized last (i.e. after digitizing all the landmarks), and they do not have to be digitized as curves, but there are advantages in doing both. One advantage of digitizing them last is that you may change your mind about where the landmarks should be placed, which would mean editing the semilandmark curves that lie between any repositioned landmarks. The other is that you will need to use **tpsDig2.16** for digitizing semilandmarks, but you may wish to use **tpsDig1.14** when you digitize the landmarks because this program can be easier to use. If you digitize all the landmarks first you will not need to switch from one program to the next as you digitize each specimen. One important feature of these programs is that the files are upwardly and downwardly compatible so you actually *can* switch between them.

Getting ready to digitize your first image:Preparing an empty data file in tpsUtil

Before you begin digitizing, it is a good idea to create a file that will contain the data. This step is not essential because you can always open an image file, digitize it and save its coordinates to a file, then do the same for the next and all remaining images, either saving each one to a separate file or appending each one to the same file. But putting each specimen in a different data file means that you will have to keep track of all the specimens that you have already digitized so that you neither forget to digitize some nor inadvertently digitize the same one multiple times. Appending each file as you go is quite risky because, when you go to save the image, the pop-up menu will give you the options to **append** or **overwrite** the file, and mistakenly clicking on overwrite will replace all the data that you have already digitized for all specimens with the data for the one specimen that you just digitized. This may seem like a difficult mistake to make, but making it only once means that you've lost all the work done to that point.

By creating an empty data file, to be filled with the data for all the specimens, you can ensure that every specimen is included in the file (and only once) and you will also be able to navigate the file and look at all the specimens in it. That can be especially useful when you need to check the consistency of your digitizing - sometimes, a landmark can migrate over the course of digitizing and landmarks are sometimes digitized out of order. You can check for digitizing error by running a principal components analysis of the data (do not worry if, at present, you do not understand what a principal components analysis is). If you run the analysis and see a single specimen very far from the others, you can go back to the file and check it. At first sight, it may look like all the others because no landmarks are obviously out of place but, if you look at the numbers next to the landmarks, you're likely to find that they are in a different order in that outlier.

The empty data file will contain the control lines for the **tpsDig** program; it will look like this:

```
LM=0
IMAGE=Aml_1508LC.jpg
ID=0
LM=0
IMAGE=Aml_1508LC.tif
```

```
ID=1
LM=0
IMAGE=Aml_1512LA.jpg
ID=2
```

(etc.)

The first line (LM=0) gives the number of landmarks, which is zero because none have been digitized yet. The second is image filename, and the third is the ID number for the specimen, which begins with zero (because the programming language C counts from zero). After digitizing a specimen, the number of landmarks is replaced by the number that you digitized and its coordinates follow that number (arranged in two columns, the *x* and *y* coordinates for each landmark). If using the scale function, the scale factor is the last line in the file.

```
LM=16
552.00000 729.00000
464.00000 914.00000
691.00000 797.00000
925.00000 808.00000
955.00000 943.00000
1064.00000 953.00000
1132.00000 962.00000
1561.00000 1032.00000
1705.00000 1329.00000
1811.00000 1182.00000
1941.00000 1137.00000
1889.00000 771.00000
1950.00000 1106.00000
822.00000 919.00000
1285.00000 644.00000
1215.00000 974.00000
IMAGE=Aml_1508LC.jpg
ID=0
SCALE=0.001515
LM=0
IMAGE=Aml_1508LC.tif
ID=1
LM=0
IMAGE=Aml_1512LA.jpg
ID=2
```

You can prepare the empty file using **tpsUtil** (which has many other useful functions that you should look at while you have it open). Start **tpsUtil** and, from the **Operation menu**, select the function **build tps file from images**; which is presently the first option on the list. Then select Input file or directory; go to the folder where you have your image files and select one of them (it does not matter which one). Then name your output file and select the format for that file (i.e. tps or NTS). We recommend tps format because virtually every morphometrics program can read it. Then click the **SetUp** button, which will give you a list of all the image files in your folder and various options (such as **Include All** or **Exclude All**). The default is to include all, but you can deselect any that you do not want to include. If you want to include just a few, you can click **Exclude All** then pick the ones that you want to include. You can choose the order of the images in the file using the **Up** and **Dn** (down) arrows or sort them (using the **Sort** function). If you select **Include path?**, the full path will be included in the filename, which is convenient when you want to keep the data file in a different folder from the images. However, doing that can complicate moving image files to a different folder (because that will change its path name) and it can also make it difficult to read the filenames in tpsDig because they will be too long to display fully.

After you have selected all the images that you want to include in that file, select **Create**. The file that you create is the one that you will open in **tpsDig**. Before leaving **tpsUtil**, you might want to produce a file that lists the image files included in the data file (you can do this now or later or both). You will want a record of the specimens in that file, especially if you begin to accumulate many data files and cannot recall which specimens are in which. You will likely want to do this again at the end of digitizing if you deleted any specimens from your file as you digitized. To produce this file, go to the last option in the **Operation** window, **List images in the tps file** go to the **Input** window and load the file that you just created, go to the **Output** window and name the file (it will be a comma delimited *.csv file), and name it with a *.csv extension, then click **Create**.

Digitizing in tpsDig

The mechanics of digitizing depend partly on which version of **tpsDig** you use, but many of the functions work the same in both versions. We will not cover all the functions that these programs have because we are not familiar with them all. If you are interested in making measurements of linear distances, angles or areas, or measuring the lengths of outlines, look at the various functions that you can find in the **Modes** and **Options** menus (and it always does help to read the Help file). The first step is to open the file that you made in **tpsUtil**, or if you did not make a file, to open an image file. In **tpsDig2.16**, go the **File** menu, select **Input source**, then **File.** If you made an empty data file, open it; if you did not make a data file, find the image format for your photos (e.g. JPG or TIFF) on the **Files of Type** list and select that, and choose the first one to digitize and click **Open**. An image should appear in the main window. You can zoom in or out using the buttons + and – on the top right of the toolbar, where you will also see a number that shows the magnification of the image.

Before you start digitizing, you may want to change the cursor. This is what you will place on the landmarks so you want to select a cursor that makes it easiest for you to locate the cross-hairs of the cursor (or the arrowhead) most precisely. Go to the **Options** menu and select **Image tools**. In the window that opens (and may immediately minimize), select the **Cursors** tab and choose the digitizing cursor that you prefer. You can try out the cursor by moving it over the image but do not click on a point unless you are on the first point that you plan to digitize; if you click, the coordinates of that point will be recorded in the file and the image tools window will close. When you find the cursor that you like best, you can select the color that you want the landmarks to be using the **Colors** tab. You can also select the size of the circle that will be used to indicate a landmark's position, and whether the circle will be closed or open, as well as the color and size of the number used to label the landmark. You can change these options at any time. Close this window and you are now ready to digitize.

To begin digitizing, find the image of the cursor (the circle with the cross-hairs) on the toolbar, or Go to the **Modes** menu, and select **Digitize landmarks,** and go to the **Options** menu, and select **Label landmarks** (this will put numbers next to the digitized points). Position the cursor over the landmark and click the left mouse button. A circle and number should appear. To digitize the next landmark just position the cursor at the appropriate point and left-click; keep going until you have finished digitizing all the landmarks that you plan to do before going on to the next specimen.

At this point, you should make a screen shot of your digitized landmarks, with visible numbers so that you will have a reminder of their locations and sequence. You can print the image and save a copy someplace convenient on your computer so you can refer to it as you digitize. To save a screenshot, go to the **File** menu and select **Save screen as**. You want to save this to a different folder (or whatever you call the screen shot will overwrite the file name for that specimen). When you are ready to digitize the next specimen, navigate back to the folder that has your files to digitize.

Saving data

You should get in the habit of saving the data regularly, preferably after every specimen. Go to the **File** menu and select **Save data**. By default, it will save the data to the tps file that you opened. If you want to change the file name, use the **Save as** option. If you did not create an empty data file before you started digitizing, and you loaded an image file rather than a data file, you will need to give a file name for the data file that you just created. Notice that the default is to save the file in the same folder as the pictures. Do not change this; when you open the file the next time, tpsDig will find and open the image file, displaying the landmarks on the image. But if you save the data file to a different folder, and if you did not include the full path name for the image files, then tpsDig will not be able to find the image file and will refuse to open the data file. To save the file, enter the file name for your data file with no extension. It will be saved with a *.TPS extension. There is no option here, despite appearances. After you have digitized the next specimen, you will have other options because you can either save its data to its own file (using the same procedure that you used for the first specimen) or you can append the second specimen's data to the file containing the data for the first specimen. To append the data, go to the **File** menu and select **Save data**. This time, when the pop-up window appears, select the existing file to which you want to add the new data, and then select **Save**. In the new window, select **Append**. The new data will be added to the end of the selected file. If you plan to append files as you go, make sure that you save a copy of the file every time that you add data to it so you do not lose all your work in case you accidentally select **Overwrite** instead of **Append.**

After you have a data file, you can edit it whenever you wish. Then, to replace the original file with the edited one, select **Overwrite** in the pop-up menu.

Using the Scale factor.

You will likely want to measure size as well as shape, so you will either need to include a scale factor in your file or else you will need to digitize two points on the ruler. Using the scale factor rather than a ruler has one notable advantage - you will not need to remove the ruler landmarks before analyzing shape in the programs in the tps series. The two points that you digitize on the ruler (or on anything else of known size) will be interpreted as if they were landmarks. And then you may find that the dominant source of variation in your sample is the position of the ruler. Not having a ruler in the file makes it far easier to check for digitizing errors. The first program in the IMP series (CoordGen) allows you to specify the landmarks that are the endpoints of the ruler, if you always digitize the ruler at the same place in the sequence (always the first two points or always the last two or always the 10th and 11th); however, it can also read a scale factor. Even so, you may wish to use the tps programs for the initial stages of data collection and analysis, and CoordGen does not produce an output file that can be opened in tpsDig (the IMAGE= line is deleted).

The problem that using the scale factor poses is that you cannot see what points you digitized to be the endpoints of a line of known length. As a result, if you see an odd value for centroid size (a topic covered in the next chapter), you will not be able to determine if that specimen is oddly sized or if you made a mistake when setting the scale. The only ways to check that are to redo the scale factor or to look at the ruler relative to the size of the specimen in the photograph and compare both to the size of the ruler and specimen size in other specimens. One solution is to use both a ruler and a scale factor - digitize two points on the ruler, and use them as endpoints when setting the scale. Then save the values for centroid size (see below) using the scale factor and the ruler; if the two files contain the same values for centroid size, you can delete the ruler landmarks (using **tpsUtil**, select **delete/reorder landmarks** from the **Operations** menu).

To set the scale factor in **tpsDig2.16**, go to the **Image tools** menu and select the **Measure** tab. You will see the default reference length of 1 mm. You do not need to change the units if you are using cm (or any other unit) instead of mm, tpsDig ignores the units. If your reference length for one specimen is 1 mm and for the next it is 1 cm; then you will need to change the numerical value from 1 to 10 when you set the scale factor for the second specimen; otherwise tpsDig will think 1 “whatever” is 10 times longer in second picture. To set the scale factor, click on **Set scale**, position the cursor on the ruler in the image and click on a point, extending the line to the other endpoint. For example, if you want your scale factor to be 10 mm, click on a point on the ruler and extend the line 10 mm, and click again. Return to the **Measure** window and click **OK**. If all your images are the same size (i.e. the ratio of pixels to millimeters is the same for each picture) you will not need to do this again; by default, the scale factor is applied to every image. If they are not the same size, you will need to set the scale for each picture, or whenever the image size changes. To reset the scale factor when you go to the next specimen, restore the **Image tools** window and set the scale factor again.

In **tpsDig1.40**, the **Set scale** function is on the **Options** menu. To set the scale, click on the ruler, extend the line to your desired length, and click on **OK**. To set the scale for the next specimen, you will repeat the same process: click on the **Options** menu, select **Set scale**, and extend the line to its desired length, the click **OK**. There was a problem with the default scaling in this version of the program, so you should use **tpsDig2.16** if you plan to use the default scaling option.

Editing digitized landmarks

Editing digitized landmarks can involve moving a landmark, inserting a landmark or deleting a landmark. In **tpsDig2.16**, to move a landmark just put the cursor on the landmark and move it (drag it to the desired location). The number will not move with it, so if you need to check that you moved the right one, go to the next image in the file (using the right arrow on the toolbar) then go back (using the left arrow on the toolbar) and check that the right number is next to the moved landmark. To insert a landmark, go to the landmark that comes after the one that you want to insert (e.g. if you forgot to digitize the seventh landmark, go to the seventh in the file). When you insert the missing landmark, the one that you clicked on will be renumbered (and all those with higher numbers will also be renumbered accordingly). This means that you can add skipped landmarks without redigitizing the entire set. Select the landmark that should be after the skipped landmark, right click, choose **Insert landmark**, and drag the new point to the correct location. Again, the number will not go with it so to make sure that you moved the inserted landmark, go to the next image and then go back. To delete a landmark place the cursor on the landmark that you want to delete, click the right mouse button and click on **Delete landmark** in the pop-up menu. Again, every landmark with a higher number will be renumbered. However, if you have resampled the semilandmarks (see below), these functions work quite differently. When you insert a landmark, the one that is added is the last one in the file (e.g. if you have 15 landmarks and click on the fifth and insert a landmark, it will be the 16th not the fifth), and the pop-up menu no longer includes the option to delete a landmark. To insert without moving all of them, or to delete a landmark, you will need to close the program, restart it and reopen the file (simply reopening the file does not work).

In **tpsDig1.4**, the same operations are used to edit landmarks, but the numbers move with the landmarks and you can always delete landmarks and you can insert them in the appropriate location (but you cannot resample the semilandmarks). If you have a fair amount of editing to do, and you want to insert a landmark between two others in all your specimens, it is easiest to use **tpsDig1.4**.

Digitizing the next specimen using Template mode

You can digitize the second (and subsequent) specimens exactly as you digitized the first. Just go to the second specimen in the data file that you created in **tpsUtil** (using the right arrow on the toolbar) or open the next image file that you want to digitize. Position the cursor where you want the first landmark to go and click on it, then go to the second landmark, etc. However, there is another way to digitize specimens after the first, which is convenient when you have many landmarks and they are not in an easily remembered order. This is to use the “template” function, which copies the landmarks from the previous specimen to the next picture. These will not be in the correct position, but they may be close enough that moving them to the correct position is less complicated than remembering which one goes where.

Template mode works very differently in the two versions of **tpsDig**. In **tpsDig2.16**, moving the first landmark will move all of them (all are translated along with the first). After that, each one can be moved to its correct location. None of the numbers move with the landmarks so it can be challenging to ensure that you have moved the right one. In contrast, in **tpsDig1.40**, moving the first landmark will move all of them, and moving the second one will rescale them. So if your images are very different sizes, moving the first one will translate all the landmarks to approximately the correct position, and moving the second will scale the entire set of landmarks to approximately the correct scale. They are only approximately correct because they are correct only for the prior specimen. After translating and rescaling them all, each one can be moved individually to the correct location and, in **tpsDig1.40**, the numbers move with the landmarks. Thus, we would recommend using **tpsDig1.40** if you plan to use the template mode.

To select the template mode, go to the **Options** menu and click on **Template Mode**. If you skip a specimen at some point in the file, template mode turns off. You can quickly do an initial round of digitizing using the template mode and adjusting just the first two landmarks in each file. Then you can return to the second and adjust the remaining landmarks. However, if your data file contains many species that differ notably in shape, you might want to adjust all the landmarks for the first specimen in each species. Then the rest of the specimens in that species will require less adjustment in the second round of digitizing.

Checking for digitizing error

After you have finished digitizing, you should do a quick check for major digitizing errors. You cannot do this until you have either finished digitizing all specimens, or have digitized the same number of landmarks for all specimens because no program will open the file if there are different numbers of landmarks on different specimens. Either wait until you have digitized all of them, or make a file that contains only the specimens that you have digitized. You can make this file in **tpsUtil**, by opening the file, saving the data to an output file (e.g. “Check1.tps”), then deleting all the specimens that you have not yet digitized. You could also open the file in a text-editing program and delete all the control lines for specimens that have not been digitized, but tps files can be very long because each landmark takes up a whole line.

The way to check for gross digitizing errors is to conduct a principal components analysis, as mentioned earlier. This will not work when there is substantial shape variation among the specimens in the file and the digitizing errors are subtle. If that is the case, you will see a scatterplot that looks reasonable even if there are mistakes in the file. The only real clue may be that locations of specimens in the scatter plot are quite different (closer together or farther apart) than you would have expected from their photographs. Thus, you will need to check that the specimens that are close in the plot really are morphologically similar, whereas those that are far apart really are quite different in appearance.

How you do this check depends on whether you used the scaling factor, a ruler or neither. If you digitized the endpoints of a ruler you will need to remove them yourself or use a program that removes them because the coordinates of the ruler will otherwise be included in the superimposition. There can be very large differences in the location of the ruler, so very large differences in “shape” could be introduced into the data simply because the ruler was above some specimens and below others. You can delete these two points in **tpsUtil**, saving the file without the ruler endpoints to a temporary file (i.e. one that is only used for this purpose). Open the program and in the **Operations** box, select **delete/reorder landmarks**, input your data file, name the output file, click on **SetUp**, and de-select the landmarks that you want to delete and finally click **Create**. Then open the principal components program in the tps series, **tpsRelw**, and load your data file. Click the buttons in the **Compute** box as they become enabled (i.e. first **Consensus**, then **Partial warps**, then **Relative warps**). To see the scatterplot, go to the **Display** box and select **Relative warps**. The points representing the specimens will not be numbered by default, so go to the **Options** menu, select Option in the pull-down list and a new screen will open with several options. On the upper right, in the **Points** box is a check-box called **Label points**. Select that, then click on **Apply** and the points in the scatterplot will be numbered. These numbers correspond to the number of the specimen in the file (e.g. the first one in the file is labeled "1", the second "2", etc.). The numbers do not correspond to the IDs, which begin at zero.

If you do find specimens that seem grossly out of place, open your data file in **tpsDig** and find that specimen; to locate it, open the **File** menu, select **Go to** and type the number of that specimen in the box (use the same number as in the plot, not the ID). Look for landmarks that are out of order, or a landmark out of position, or damage to the specimen. Subtle digitizing errors (and subtle damage) are rarely detectable this way, but you may be able to discern inconsistencies in your digitizing. For example, if all the low numbered IDs are on one side of the plot and all the high numbered IDs are on the other, you might want to check whether that ordering is expected given the order of the specimens in the file. If a gap between high and low numbers or progression between them is not anticipated, check for landmarks whose position was inadvertently redefined over the course of digitising - especially if a gap corresponds to different days of digitizing.

Digitizing semilandmarks

When you digitize semilandmarks, remember that these will be turned into landmarks. Therefore, the order in which you trace the curves, and the direction in which you trace each one of the curves, matters. For example, if you first trace the curve between incisor and molar on the jaw of the first specimen, that same curve must be the first curve in all other specimens. Similarly, if you start tracing at the incisor and end at the molar, you must always start at the incisor and end at the molar. If you do not, when these semilandmarks are turned into landmarks, the first landmark in the first curve of one specimen will not correspond to the first landmark in the first curve in another specimen. You will want to trace the curves continuously (to avoid the risk of producing two curves instead of one). So before you start tracing a curve, make sure that the whole curve is visible (reduce the image if it is not). You also want to see the curve clearly so you can trace it as closely as possible, so do not reduce the image by too much. Once you can see the curve clearly, select the curve-tracing (yellow pencil) tool on the tool bar. To trace the curve, just start drawing it. Left-click on a point along the curve, then on another, and you will see a line appear between the two points. Keep going by moving to the next point on the curve, left-clicking as you go. Do not worry about the number of points that you have digitized because you will tell **tpsDig** how many points you want when you finish the curve. When you get to the end of the curve, right-click. Now you will set the number of points in the curve; with the cursor over the curve, right click and **resample** the curves. The default is the "by step" option, but you will probably want the **by length** option so select that (this permanently changes the option until you change it again). Type the desired number of points in the box.

Occasionally, you may find that you started to digitize the curve even though you could not see both ends of it; this is not a fatal mistake. You can extend the curve by grabbing one end of it and just pulling it to the endpoint. Then, put the pencil tool over the curve and left-click to add a point, move that closer to the curve, left-click again to add another point, reposition the curve again, and keep going until the curve you have traced lines up with the curve of the specimen. If the curve is fairly straight, this will not take many additional points. If it is very complex, it will take many (and if you find this a cumbersome way to fix the curve, you could always delete the curve and start it again).

Once you have a curve with the desired number of points, see how well it matches the curve of the specimen. If there are too few points to reconstruct it accurately, choose a larger number of points (go back to the "resample" function and increase the number). If there are so many points that they are all bunched up and look uneven, reflecting minor twitches of your hand instead of the curvature of the specimen, choose a smaller number of points. Selecting the appropriate number of points when your sample includes a diverse array of morphologies can be difficult if they differ in how long their curves are. You may find that the semilandmarks bunch up in some but not in others. In this case, it is better to keep the larger number because you can always reduce it later.

If you see that some points (or segments of the curve) do not closely match the specimen's curve, you can edit individual points by moving them - just hover over the curve until you see the points, then move one (or more). You can also delete semilandmarks; right click when you have the cursor on the curve point and select **Delete curve point**. But be very careful - this option is the first one on the list when you are directly over that point when you right-click, but if you are not directly over a point, the first option on the list is **Delete *curve***. You will not be the first person to delete a curve accidentally. If you have deleted the last curve you digitized, this is not a serious problem because you can just redo it. However, if you are editing the first curve and you accidentally delete that one, you have two unpleasant choices because if you redo the first one after you digitized all the others, the curve that ought to be first will instead be the last one in your file - you digitized it last. The first choice is to delete *all* the curves that come after the one you accidentally deleted, and redo all of them, the other option is to close the file without saving it and reopen it. Using that second option will cause you to lose any work that you did since you last saved the file. Need we remind you to save your data regularly? After moving or deleting points, go back to the resample function so that they remain evenly spaced.

If the curve crosses the midline, treat it as two symmetric curves that meet at midline. We recommend that you anchor each curve by landmarks at both ends; if you begin a curve in the middle of a landmark you will have similar spacing between the landmark and the first semilandmark as you do between semilandmarks. However, if you do this, you will need to delete the first and last semilandmarks because, if they actually have the same coordinates as the landmarks, the next program you use may crash. It is OK to have two curves anchored by the same landmark(s).

Literature Cited

Chiari, Y., Wang, B., Rushmeier, H. & Caccone, A. (2008) Using digital images to reconstruct three-dimensional biological forms: a new tool for morphological studies. *Biological Journal of the Linnean Society* **95**: 425-436.

Dhondt, S., Vanhaeren, H., Van Loo, D., Cnudde, V. & Inzé, D. 2010. Plant structure visualization by high-resolution X-ray computed tomography. *Trends in Plant Science* **15**: 419-422.

Hallgrímsson, B., Zelditch, M. L., Parsons, T. E., Kristensen, E., Young, N. M., Boyd, S. K. (2008) Morphometrics and biological anthropology in the post-genomic age. In: *Biological Anthropology of the Human Skeleton*, (Katzenberg, M. A. & Saunders, R., eds). pp. 207- 236. Wiley, New York.

Rohlf, F. J. (2010) tpsDig. pp. Department of Ecology and Evolution, State University of New York, Stony Brook.

van der Niet, T., Zollikofer, C. P. E., de Leon, M. S. P., Johnson, S. D. & Linder, H. P. (2010) Three-dimensional geometric morphometrics for studying floral shape variation. *Trends in Plant Science* **15**: 423-426.

3

Superimposition

This chapter covers three procedures that prepare the digitized coordinates for a shape analysis: (1) superimposition of landmarks and semilandmarks, (2) "symmetrizing" the data (i.e., reflecting and averaging bilaterally symmetric landmarks and eliminating redundant bilateral homologues) and (3) estimating coordinates of missing landmarks from those on the other side of a symmetric object. The second and third procedures are clearly related to each other, but there are two kinds of symmetry that need to be distinguished because only one offers a straightforward approach to estimating coordinates of missing landmarks. The two kinds of symmetry are object and matching symmetry (Klingenberg et al., 2002). Briefly, object symmetry refers to the case in which both sides and the midline are present in the same configuration (Fig. W3.1A) whereas matching symmetry refers to the case of two configurations,

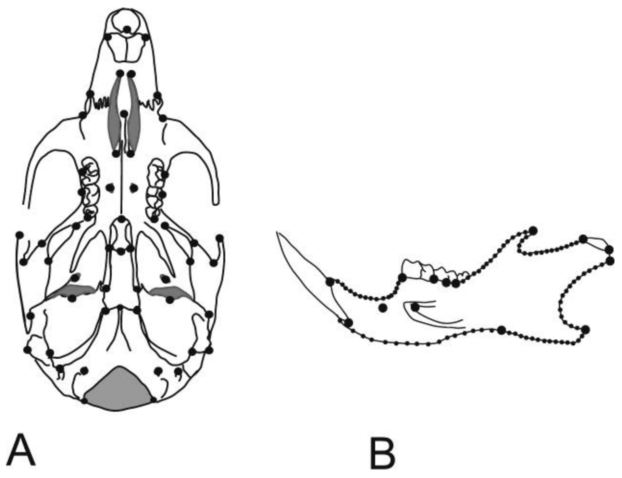


Figure W3.1. Two kinds of bilateral symmetry. A: Object symmetry; B: Matching symmetry.

one for the left side and one for the right (Fig. W3.1B). The topic of (a)symmetry is an important one for evolutionary developmental biology and we return to it in much greater depth in Chapter 12. For now, we are only concerned with symmetrizing our data, whether they exhibit object or matching symmetry, and estimating coordinates of missing landmarks when data exhibit object symmetry. Methods for estimating missing landmarks on objects that lack bilateral symmetry are not as well established so we do not cover them in this workbook.

How exactly you go about preparing your data for a shape analysis depends on whether (1) you have only landmarks or also have semilandmarks, (2) the kind of symmetry in your data and (3) any landmarks are missing. What complicates the procedure is the fact that the programs that can symmetrize the data cannot perform a semilandmark superimposition (conversely, those that can perform the semilandmark superimposition cannot symmetrize the data), and that none of the standard morphometrics programs can read files with missing data in them. Thus, if you want to symmetrize landmarks, you will use one program, but if you want to symmetrize semilandmarks you will use (at least) two, and if you want to estimate the coordinates of missing landmarks and superimpose them you also will use two programs, but not the ones that you would use for symmetrizing landmarks when none are missing.

To organize the flow of your work (and this chapter) we distinguish six cases. You have: (1) only landmarks, none missing and your data are not bilaterally symmetric, (2) only landmarks, none missing and your data *are* bilaterally symmetric; (3) landmarks and semilandmarks, none missing, and your data are not bilaterally symmetric; (4) landmarks and semilandmarks, none missing, and your data *are* bilaterally symmetric, (5) landmarks, some missing, and your data are bilaterally symmetric; (6) landmarks and semilandmarks, some missing, and your data are bilaterally symmetric. For these six cases, we first list the procedures that need to be done and the programs that can do them. After that, we describe the programs and how they carry out these procedure(s). Because some programs carry out more than one (e.g., **CoordGen** superimposes landmarks and semilandmarks and **MorphoJ** superimposes landmarks and symmetrizes the data) we discuss multiple procedures when we describe those programs. As you go through the following section, figure out which procedures you need to do, then find the section that covers how to do it. If a single program does both procedures (no one of them does all three) then you can use the same program for both. The two programs that do both a landmark and semilandmark superimposition are discussed twice and the first time we cover only the landmark superimposition.

(1) You have only landmarks, none missing and your data are not bilaterally symmetric:

You have only one procedure to do, a Procrustes superimposition. Procrustes superimposition of 2D data can be done in many programs (including **CoordGen** (IMP), **tps** programs, **MorphoJ**, **shapes** (and the **R** functions written by Claude) and **PAST**. Procrustes superimposition of 3D data can be done by **Simple3D** (IMP), **MorphoJ**, **shapes** (and the **R** functions written by Claude) and **PAST**.

(2) You have only landmarks, none missing, and your data are bilaterally symmetric:

You have two procedures to do, but, with one exception, the same program can do both. Programs for estimating the symmetric component of shape include: **MorphoJ** (2- and 3D data), **Sage** (2D data).

(3) You have landmarks and semilandmarks, none missing, and your data are not bilaterally symmetric.

You first need to do a semilandmark superimposition (the landmark superimposition is done as part of this procedure). Programs that can do a semilandmark superimposition of 2D data are **tpsRelw** and **Semiland**, a module within **CoordGen** (IMP). **ChainMan3D** is the only freely available program, to our knowledge, that can superimpose 3D semilandmarks, but it is restricted to semilandmarks along curves, meaning it cannot superimpose semilandmarks distributed across surfaces. Using it may require editing your data file to insert "break points" (see below). There are programs that will superimpose semilandmarks on surfaces: **Edgewarp** (which runs on Linux and Mac OS; if it is now included in **EVAN**, it should run on Windows). Other methods (and platform independent versions of them) should be available soon; when they are, they will be included in the updated workbook.

(4) You have landmarks and semilandmarks, none missing, and your data are bilaterally symmetric.

You need to do the semilandmark superimposition (in **tpsRelw**, **CoordGen** or **ChainMan3D**), then symmetrize the data in **MorphoJ** or **Sage**.

(5) You have only landmarks, some missing, and your data are bilaterally symmetric.

You first need to estimate the coordinates for the missing landmarks before your file will be read by a program that can superimpose the data. The program that can estimate the coordinates for missing landmarks from their bilateral homologues is **OSymm()**, by Annat Haber, (http://home.uchicago.edu/annat/homepage/R_scripts_files/OSymm.R, also available on the Stony Brook "Software" page, "Programs for R"). **OSymm()** uses the optimal method for estimating object symmetry following Klingenberg and colleagues (2002) and can analyze both 2- and 3D data. After running **OSymm(),** and perhaps reorganizing the output to put the landmarks in the same order as the others in your file, you can superimpose the (already symmetrized) landmarks.

(6) You have landmarks and semilandmarks, some landmarks missing, and your data are bilaterally symmetric.

This is the difficult case. You should to do the semilandmark superimposition first so that the arbitrary variation in semilandmark spacing does not influence the estimate of the symmetric shape. But none of the programs for semilandmark superimposition can read a file with missing data in it, and **OSymm(),** the program that can estimate the coordinates of missing landmarks does not do a superimposition (landmark or semilandmark). You have two options. One possible way to proceed is to put a missing landmark near to its proper location (not off in a corner, where it could heavily influence the superimposition). Then do the semilandmark superimposition, and when that is done, use the method above for imputing a missing landmark (**OSymm()**). The alternative is to ignore the arbitrary variation in the semilandmarks, treat them as if they are landmarks, use the procedure above for imputing missing landmarks and then do the semilandmark superimposition. The difference between these two approaches is whether you do a semilandmark superimposition first, followed by estimation of the missing landmarks, or instead, do the estimation of the missing landmarks first, followed by the semilandmark superimposition.

Programs for landmark superimposition

In this section, we cover several programs that do a Procrustes superimposition of landmarks. You will want to see the results, if only to check for mistakes, so we explain how to do the superimposition first and then how to get output graphics that could alert you to outliers or digitizing mistakes. If your data are 2D, and you used the **tpsDig** scale factor when digitizing, you can use nearly all the programs discussed in this section (**CoordGen**, **tpsRelw**, **MorphoJ**, **shapes** and **PAST**). However, if your data are 2D and you used the endpoints of a ruler, you should use **CoordGen** because the other programs will read the coordinates of the ruler as landmarks. If your data are 3D, you can use **MorphoJ**, **shapes**, **PAST** and **Simple3D** (IMP). Because the operation of **MorphoJ**, **shapes** and **PAST** are the same for 2- and 3D data, we cover them once. Two of the programs discussed in this section superimpose semilandmarks as well as landmarks (**CoordGen** and **tpsRelw**) but if you have semilandmarks, go directly to the next section *(Semilandmark and landmark superimposition*).

CoordGen (2D)

To superimpose landmarks in **CoordGen,** load your datafile according to its file format. Four options for datafile formats are given in the orange **Load Data** box, and another for landmarks is on the menu in **File Options** (the others there are for semilandmarks). The first two options in **Load Data** are for files in tps format, the top one is for a tps file that has a ruler in the file (so two of the landmarks are not really landmarks but are instead the endpoints of the ruler). The one below that is for a tps file with no ruler or scale factor. If, as is most likely, you have a tps file with a scale factor, go to the **File Options** menu; it is the topmost option there. The other three options in **Load Data** are for more conventional datafile formats: each individual's data is in a row, with the *x*-coordinate for the first landmark preceding the *y*-coordinate for that landmark, then the *x*-coordinate for the second coordinate preceding the *y*-coordinate for that landmark, etc. The first option contains a ruler in the file, the second does not, and the third is for a file in IMP format, with centroid size in the last column. It is labeled as a "BC File" because **CoordGen** can save Bookstein Coordinates plus centroid size. If you have a ruler in your file, **CoordGen** will guess that its endpoints are the last two landmarks in the file. If that is not correct, put the correct landmarks in the blue **Carry Out Rescaling** box and press the button.

When the file is loaded (or the rescaling is carried out), **CoordGen** will do a Procrustes superimposition. You will see the landmarks for your data in blue and their means in red. If it is not oriented the way that you like, you can reorient it (and the coordinates that you will save) by first producing Bookstein shape coordinates and then saving the Procrustes coordinates. To produce Bookstein shape coordinates, enter the desired endpoints of the baseline in the blue **Baseline** (for *MouseSquirrel1.tps,* try landmarks 11 and 2). Then go to the green **Display Box** below, and click on **Show BC**. You can then **Show Procrustes**. If you want to see what a sliding-baseline superimposition looks like you can **Show SBR** and similarly if you want to see resistant-fit superimposition looks like, you can select **Show RFTRA**. The figure can be saved, with or without the axes, by clicking on **Copy Image to Clipboard** or **Print Image to File** (the options are *.TIFF, *. JPEG, uncompressed *.TIFF, *.PNG, *.EPS and *.SVG). If you want to remove axes from the image, click on **Clear Axis**. The **Numbers on Landmarks** option will display the number of each landmark on the image (near the mean location of that landmark in the datafile). The **Figure Options** pull-down menu has several other options for controlling the image, including changing the size of the symbols and filling them.

The program saves the coordinates in two datafile formats. Find the **Output File Format** (above and to the right of the deep blue "**Save Coordinates**" box). The default is the IMP format (X1Y1…CS), but you can change this to tps format by clicking on that button. After choosing the format, go to the blue **Save Coordinates** box and select the type of coordinates you wish to save (the fifth on the list is Procrustes shape coordinates). There are several other options on the **File Options** menu. One is the "**Save Procrustes Data, tab delimited, labels and Header**". This saves a tab-limited file, with labels for the coordinates in the header row, and, in some cases, the file names on the IMAGE= line from the t**psDig** file are in the first column. Sometimes the filenames for a subset of the specimens are in the first column, sometimes none of them are.

There are two functions on the **Call Other Tools** menu on the toolbar that are useful for checking for digitizing error, although these work best if you have digitized multiple replicates of the same individuals and the individuals are from a single homogeneous population. Otherwise the results will still be informative about the variation in your sample but that variation will not be due largely to digitizing error. One function shows the variation contributed by each landmark as inferred by removing them one by one and looking at how much the variance drops after that removal. Another function does a quick principal components analysis (PCA). This function does not replace a dedicated program for PCA, it merely provides a scatter plot that you can check for gross outliers or other potential problems in the file. To see the contributions of the landmarks to the variance, pick the last option on the menu, **LM Contributions to Variance**. What you will see in the graphics window is a histogram, with the variance on the *y*-axis and the frequency of the landmarks on the *x*-axis. If you loaded *MouseSquirrel1.tps*, you will see that landmark 8 has the greatest impact on the variance, followed by landmark 14. A pop-up window will appear and show you the variance in the file when a landmark is removed. The top one on the list is the one whose removal most decreases the variance, followed by the next, ending with the one whose removal has the least impact on the variance. To see the PCA plot, pick the third option, **Quick Diagnostic PCA**. If you are working with *MouseSquirrel1.tps*, you will see a very large gap between small clouds of points, the deer mice are the first 10 in the file, the fox squirrels are the last 10.

There are other functions in **CoordGen**, such as calculating the mean shape in a file, or the mean of the smallest or largest specimens, but these are not yet relevant.

To load a new data file, use **Clear and Reset** to remove a data set from the program and go back to **Load Data** to load a new one.

tpsRelw (2D)

To superimpose landmark data in **tpsRelw**, open the program and load your datafile (which must be in tps or NTS format). At this point, only one button is enabled: **Consensus**; click on it. That function did the Procrustes superimposition (and calculated the mean shape, i.e., the consensus). To save the superimposed coordinates and centroid size, go to the **File** menu, select **Save**, and select **Aligned specimens** in the pop-op window, name the file for the saved coordinates and select the desired datafile format (tps or NTS). Then return to the **File** menu, and again select **Save**, and then select **Centroid size** (this is saved in NTS format). To see the mean shape, go to the **Display** options (on the right side of the interface) and click on **Consensus**. To check for gross outliers (or other problems in your datafile), continue the analysis (on the left side of the interface). Select **Partial Warps** (the subject of Chapter 5) and then **Relative Warps** (which are principal components because that is the default option in this program). Then go to **Display** and select **Relative Warps** and you will see the scatter plot, labeled by the specimen numbers (not their IDs, the first specimen in the plot is "1", not "0"). The plot might not look exactly like the one that you got from **CoordGen** (the *x*-axis might be flipped horizontally, the *y*-axis flipped vertically, or both). As you may recall, the sign of the principal components is arbitrary.

MorphoJ (2- and 3D)

When you went through the process of loading your files in Chapter 1, you created a project in **MorphoJ.** If you used MouseSquirrel1, your data do not have object symmetry, so we first go through the procedure for superimposing shapes without object symmetry, then discuss the procedure for symmetric objects. Open the program, go to the **File** menu, and, if you saved your project when you loaded your data before, select **Open Recent Project**. Otherwise, load your data according to the procedure discussed in Chapter 1. Go to the **Project Tree** tab, where you will see the dataset you loaded. Select it, and go to the **Preliminaries** menu and select **New Procrustes Fit**. You can either align the data by the principal axes of the mean shape or you can select two landmarks to align the mean (if using MouseSquirrel1, select landmarks 11 and 2; you have to type in the "L" for landmark, not just the number of the landmark (it does not need to be uppercase). You can change the two landmarks, or switch to using the Principal Axes. To see what the mean looks like, select the datafile (which now contains the superimposed coordinates), right click and select **Display Graphs**. **MorphoJ** has a check for outliers on the **Preliminaries** menu, but we've used PCA for this purpose to this point so we'll use that same method in **MorphoJ**. To do the PCA, select your dataset, then go to the **Preliminaries** menu and **Generate a Covariance Matrix**. The pop-up window will open, and the data set you selected will be highlighted; the **Data types** available for that dataset will be listed (you only have **Procrustes coordinates** at this point). Select that file then click **Execute**. Then go back to the **Project Tree**, select the covariance matrix that you just generated and then go to the **Variation** menu and select **Principal Components Analysis**. **MorphoJ** will go to the first picture in its **Graphics** tab labeled **PC shape changes** – what **MorphoJ** calls a "lollipop graph" with vectors extending from landmarks but without arrowheads. We will explain this graph and the options for modifying it in Chapter 5. For now, click the tab labeled **PC scores** to get a scatter plot, which should look familiar by now. Right click on the graph to get a menu for modifying the plot, and check **Label Data Points** to have identifiers written on the plot.

If your data have object symmetry, you should specify that when you load the file. **MorphoJ** will infer which landmarks are paired with which and, in our experience, it is always correct. If not, you can correct the pairings. When you request a superimposition, **MorphoJ** will symmetrize the shapes by copying each one, reflecting it on the original configuration across the midline, superimposing them and computing the average of the original and reflected Procrustes coordinates. You will thus get two data sets and plots of the symmetric and asymmetric components. When you go to generate the covariance matrix, the pop-up window will then ask you whether you wish to generate the covariance matrix for the symmetric or asymmetric shapes.

If your data are 3D, just say so when you create your data set. If your 3D data have object symmetry, **MorphoJ** will infer which landmarks are paired with which, and when you request the superimposition, it will produce the symmetric and asymmetric components and show three 2D plots for each, one in the *x*,*y*-plane, another in the *x*,*z*-plane and another in the *y*,*z*-plane.

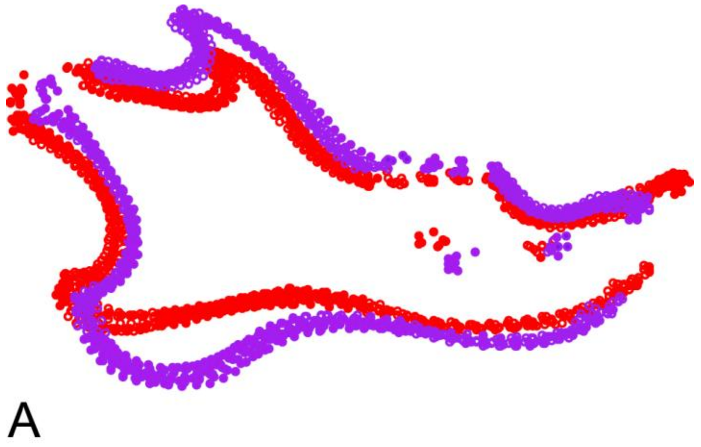
shapes (2- and 3D)

Open **R** and then load your datafile using one of the functions that reads tps datafiles (**read.tps1**, **read.tps2** or **read.tps3**). Go to the **File** menu, select **Open Script** and navigate to the folder where you put the **read.tps** functions to open the one that you want. It will open in a script window. In that window, right click and **Select all** and then right click again and **Run line or selection**. Or you could just copy the script into the Console window at the command prompt (>). Now you need to run it on your data. Type: "`data <-read.tps(file.choose())`", which will open a window and let you navigate to your data file. Select your data file. The output of the function will be saved into the variable *data*. You now have a datafile in the format for **shapes**.

You then need to load the **shapes** package, which has the function we will use to do a Procrustes superimposition. To load it, you can use either of two functions, **library ()** or **require(). Require** checks to see if you've already loaded the package. In the **Console** window, at the command prompt, type either "`library(shapes)`" or "`require(shapes)`". You can now run any function in the **shapes** package. The one that we use for a Procrustes superimposition is **procGPA**. We will assign the output of that function to "`super`" (you could call it anything that you like). So at the command prompt, type: "`super <- procGPA (data)`". You will not see any output, but it is there in *super*; to see the results, type "`super`". As you will see, there is a lot of output, from coordinates (*super$rotated*) to centroid size (*super$size*) to the mean shape (*super$mshape*) to principal component scores (*super$scores*). You can work with the superimposed coordinates or any of the other outputs. For example, we will plot the superimposed coordinates and color-code our results for deer mice and squirrels, but to do that we need to tell **R** which specimens are deer mice and which are squirrels. The first 10 in the file are deer mice and because they are now in the **shapes** format, they are arranged in an array; the third index refers to the specimen number. So the data for the first 10 specimens are specified by the indices enclosed by square brackets: [,,1], [,,2], [,,3] to [,,10]. To specify the whole set, we can write: [,,1:10]**.** The next 10, [,,11:20] are squirrels. We can put the deer mice into one dataset and the squirrels into another. That is not necessary, but doing so makes typing easier. Then we "**plot**" the *x*- and *y*- coordinates, and we say what we want the graph to look like. So, the whole procedure, from doing the superimposition of our data file, “data” to plotting the color-coded landmarks is this:

```
        super<-procGPA(data)
        mouse<-super$rotated[,,1:10]
        squirrel<-super$rotated[,,11:20]
        plot(mouse[,1,],mouse[,2,],asp=1,xlab="",ylab="",
        cex=2.8,axes=F,pch=20,col="red")
        points(squirrel[,1,], squirrel[,2,],pch=20,cex=2.8,col="purple")
```

We plotted deer mice coordinates in red and squirrel coordinates in purple (Fig. W3.2A). To see what **cex** and **pch** and even **asp** mean, try changing the values and look at what happens to the plot. Now



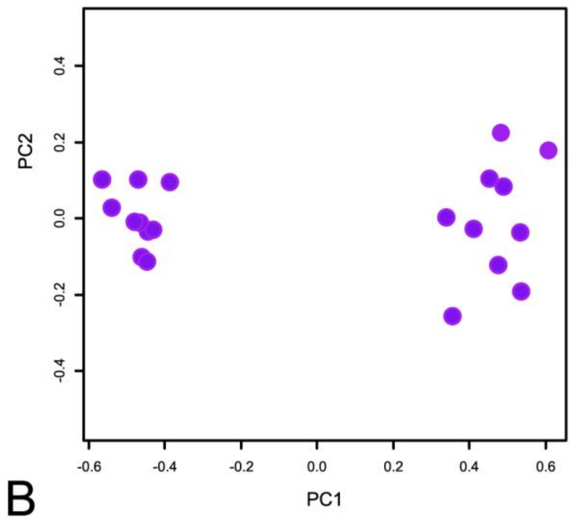


Figure W3.2. Superimposition in R. A: Plot of superimposed coordinates for deer mice and squirrels; deer mouse landmarks shown in red, squirrel landmarks shown in purple. B: Scatter plot of principal component scores.

that we have the plots we can look for outliers in the distribution of the landmarks. But we also obtained the principal component scores when we ran procGPA so we can look at the scatter plot of scores on the first two PCs when seeking outliers. These are contained in the first two columns of super$scores (they have the indices [,1] and [,2]):

```
plot(super$scores[,1],super$scores[,2],asp=1,xlab="PC1",ylab="PC2",cex=3.5,
pch=20, col="purple")
```

That gives us the scatter plot (Fig. W3.2B). If there were any outliers, we would go further and label the plot with specimen numbers (or color-code the individual data points) to see which ones are the outliers.

PAST (2- and 3D)

You loaded your data into PAST when we covered loading data in Chapter 1. If you saved your file then, you can open it in PAST (otherwise, load it now). To do a Procrustes Superimposition, first select all your data (highlight it). Then go to the **Transform** menu (on the toolbar on top). Select **Procrustes 2/3D**; when the window pops up, select the dimensionality of your data (2D if using MouseSquirrel1). The default is to rotate to the principal axis of the mean (consensus) so if you want to use that option, click **OK**. Once again we will do a PCA to look for outliers. Select your data, go to the **Geom** menu (on the toolbar) and select **Relative warps** (fifth from the bottom of the list). When the window opens, select **Rel. warp scores**. The plot may look unusual because of the way that the axes are scaled but if you maximize the window it will look like the plots you made in the other programs.

Simple3D (3D)

Simple3D is designed for multiple purposes so the interface is rather complicated. Simple3D superimposes two files of 3D landmarks, does a statistical comparison of the two means and also of the variances. The program can, however, be used to superimpose a single file of 3D landmarks. The file can be in three datafile formats (1) tps format (with three columns of coordinates following the LM = line (but not LM3 =), (2) an IMP-like format with each individual's data in a row (*x*-coordinate, then *y*-coordinate, then *z*-coordinate for each landmark), but lacking centroid size and (3) IMP format, which is the same as (2) but with centroid size in the last column. The program considers both the second and third "IMP format". You only "unselect" IMP format if you have tps format. If you have the format with centroid size in the last column (as you would if you were reloading coordinates that you had already superimposed), you need to check that **CS value in input** is on; (it is below **IMP format input files**, which is below the red-orange **Load Data 2** box). If centroid size is not in the input, make sure that **CS value in input** is off.

After loading the file(s), save the output file (on the Load file menu, below "Show Mean (1)" and "Show Data" is the button **Save X1Y1Z1CS** file, which means that the output file format has each individual in a row, with the *x*-coordinate followed by the *y*-coordinate followed by the *z*-coordinate for each landmark, and centroid size in the final column). To calculate centroid size, when it is not present in the input datafile, the program presumes that all the images are correctly scaled when they are loaded into the program. To see the superimposed coordinates, go to the graphics menu in the yellow-green box on the left, click on **Open 3D Figure** and a graphics window will open. You can rotate the data using the Rotate 3D tool (on the right of the hand). You can save the figure in a variety of graphical formats using the **Save Figure** button. You can also insert a legend or color bar; if you cannot remove them after you inserted them, you can close the graphics window, then reopen it and **Show Data** again.

Programs for semilandmark and landmark superimposition

Two programs do a semilandmark superimposition of 2D data, the **Semiland** module within **CoordGen** and **tpsRelw**. We will call the **Semiland** module "**CoordGen"** because you will start the process in **CoordGen** and invoke "**Semiland**" as a tool within it. Only one freely available program that runs under Windows will do a semilandmark alignment of 3D data (**ChainMan3D**). This program is so different from the others that we discuss it separately, below. Before either **CoordGen** or **tpsRelw** can do a semilandmark superimposition, you need to prepare the datafile, by removing the control lines that **tpsDig2**.**16** inserted to distinguish landmarks from background curves (i.e., CURVES = and POINTS =), You also need to prepare a file that tells the program which points are semilandmarks and which points they slide between. **CoordGen** calls this a "protocol" and **tpsRelw** calls it a "sliders" file. **CoordGen** can prepare both these files for you, but it will not inform you of mistakes in your file nor will it necessarily produce the protocol that you want to use. We recommend using **CoordGen** for these two functions only if (1) you are certain that there are no mistakes in your datafile, (2) you will not want to delete any semilandmarks (such as those that overlap landmarks), and (3) you do not want the semilandmarks to be anchored by landmarks at the two ends of the curve. If there are mistakes in your datafile, **CoordGen** will not alert you to them, nor recognize them—it will just misread the data. And the protocol file that it makes does not include the landmarks at the ends of the curves. In general, the safest way to proceed is by using **tpsUtil** for preparing the datafile and correcting your mistakes in **tpsDig** as they are found. You can use a text editor (or spreadsheet) to prepare the protocol file for **CoordGen** or **tpsUtil** to prepare a sliders file for **tpsRelw**.

Removing the control lines in tpsUtil

Before doing anything else, save a copy of your original datafile with a new name (so that you don't overwrite it as you convert the curve points to landmarks). You may decide later that you want to add more curves, move one of the landmarks that anchors a curve (and therefore all the semilandmarks on it) or increase the number of semilandmarks on one or more of the curves. Once you have saved a copy of the file, open **tpsUtil**, and on the **Operation** menu, select **append tps Curve to landmarks.** *Do not select* "Convert tps XY outlines to landmarks"—that will delete all the landmarks from your file! Select your input file, and name the output file (e.g., MouseSquirrel1Edited.tps). You can open that same file in **tpsDig2.16** (this will not produce an I/O error). In **tpsUtil**, click **Create**. As **tpsUtil** goes through the input file it will alert you to specimens that have the wrong number of semilandmarks or the wrong number of curves. When the number of semilandmarks on a particular curve does not match the number in the previous specimen, **tpsUtil** will tell you which curve, on which specimen, does not match the number of the previous specimen. If you have your input file open in **tpsDig2.16**, you can go to the specimen that has the wrong number of semilandmarks (using the **Go to** function on the file menu), correct your mistake and save the corrected data (to the same file). Then, when you click **Create** again, **tpsUtil** will progress to the next specimen to be corrected. When **tpsUtil** finds a specimen with the wrong number of curves, it will not tell you which one it is (just that there is a specimen with a different number of curves). You can scroll through the file looking for a specimen that is missing a curve. Even in a large file, this is usually pretty obvious. If all that is missing is the last of the curves, you can simply add it. Unfortunately, if the missing curve is not the last one, you cannot just add it (it would then be the last curve that you digitized, making it out of order). You will need to delete all the curves that were supposed to be after the missing one and redo them all. In some cases, it will not be visually obvious that there is a mistake. Sometimes, you started and ended and restarted a curve, so that there are two where it looks like there is one. If you cannot see any mistakes, exit **tpsDig2.16** and open your datafile in a text-editor and search for "Curves =". When you find a specimen with the wrong number of curves, look for the curve that has the wrong number of points on it. When you find that, note the ID for the specimen so you can find it in **tpsDig2.16** (remember that the IDs start with zero but the **Go to** function counts from one). Check for any other specimens with the wrong number of curves before reopening your file in **tpsDig2.16**. Go to the specimen(s) that you need to correct. If curves are split into two, delete the one that has the fewer points. Extend the other curve to so that it begin and ends at the correct point (drag the last of the curve points to the proper starting or ending position), and "resample" it so that it has the correct number of points. After correcting that mistake, save the file and return to **tpsUtil**. When all mistakes are corrected, clicking **Create** will create your edited file.

Before leaving **tpsUtil**, you can delete any semilandmarks that overlap the landmarks if you began and ended your curves at landmarks. Select the Operation **Delete/reorder landmarks** and input the file that you just created and name the output file (e.g, MouseSquirrel1Overlap deleted.tps). Go to **Set Up,** and select the landmarks that you wish to delete and **Create** that file. You can open it now in **tpsDig** to make sure that you deleted the right landmarks (this is also a good time to delete any unnecessary files because they can accumulate rapidly).

Creating the "protocol"/ "sliders" files.

The protocol file for **CoordGen** and the sliders file for **tpsRelw** have different formats; we first describe producing the protocol file for **CoordGen**, then the sliders file for **tpsRelw**. **CoordGen** will produce the protocol file if you input a file with the tps control lines in it (these are the control lines that we just removed in **tpsUtil**). You could go through the procedure for removing the control lines, correcting your file as described above, but avoid creating the file without the control lines so that you have a corrected file with all the control lines in it. One way to do this is to save your **tpsDig2.16** file to a new name each time you correct it, opening that corrected version in both **tpsUtil** and **tpsDig2.16** until **tpsUtil** creates the file without the control lines. Then you revert to the version of the datafile that had only one more error in it and correct that one, without creating the file in **tpsUtil**. You will not be able to delete any semilandmarks that overlap landmarks until later, when the curves are turned into landmarks. Alternatively, you can make the protocol file in a text-editor or spreadsheet.

To make the protocol file in a spreadsheet, you will need four columns. The first identifies the curve on which the point is located. Landmarks are on curve "zero". The first curve is designated as "1", the second as "2", and so forth. If you have a "closed curve", meaning that the first and last points are adjacent to each other, that is indicated by a "-" before the curve number (e.g., -1). The first closed curve is -1 even if it is not the first curve in the file. The second column is the ordinal position of a point along the curve; the first point on a curve is "1", the second point is "2" and so forth. When you want to anchor your curves by landmarks, the landmarks are the first point on a curve. Curves can begin with landmarks, even though the landmarks were all listed on curve zero. That anchoring landmark will thus appear on the zeroth curve and on any other(s) that it anchors (the same landmark might anchor two curves). The third column is the ordinal position of a point in the tpsDig file; the first point in the file is "1", the second is "2" and so forth. The fourth column says whether the point is a landmark ("0"), or a semilandmark ("1") or a "helper" ("2"). Helpers are points that are used for aligning the semilandmarks but are removed from the datafile after the alignment is done. In MouseSquirrel1, for example, there are many more points on each curve than we might want in our final datafile—they are included solely for the purpose of aligning the semilandmarks. Any semilandmarks that overlap landmarks would be designated as helpers so that they get removed from the datafile after the alignment.

A fragment of a semilandmark protocol file is shown in Table 1. There are 12 landmarks on curve zero. There are six points on curve 1, the first of which is landmark 3, the last of which is landmark 4, and the semilandmarks between them are the 13th, 14th, 15th, and 16th points in the tpsDig file. The 14th point in the tpsDig file, which is the third point on the first curve, is designated as a helper.

Table 1. A fragment of a "Semilandmark Protocol" file for CoordGen. The rows with zeros in the first column are the landmarks, the rows with 1 in the first column are the points on the first curve. The second column is the number of a point along a curve, the third is the number of the point within the digitizer file, e.g., tpsDig file. The last column identifies the point as a landmark (0), semilandmark (1) or "helper" (2). Note that there are two landmarks, 3 and 4, at the ends of the curve.

0	1	1	0
0	2	2	0
0	3	3	0
0	4	4	0
0	5	5	0
0	6	6	0
0	7	7	0
0	8	8	0
0	9	9	0
0	10	10	0
0	11	11	0
0	12	12	0
1	1	3	0
1	2	13	1
1	3	14	2
1	4	15	1
1	5	16	1
1	6	4	0

The sliders file for **tpsRelw** can be made in **tpsUtil**. On the **Operations** menu, select **Make sliders file**. The input file is your datafile (with all the control lines removed) and the output file is your sliders file (which will be in NTS format). When you click **Set Up** you will see a picture of all your landmarks and semilandmarks, all colored red (Fig. W3.3A). You make the sliders file by dragging the cursor through three points. The second slides between the first and the third. So start the cursor on the first point, such as a landmark anchoring the semilandmark curve, and drag through the first semilandmark to the second. If you succeed, the second point (first semilandmark) will now be white (Fig. W3.3B) and your computer will beep, assuming the sound is on. If that doesn't happen, try again. You then drag the cursor from the second through the third to the fourth point (overlapping two of the points from the previous step), and the third will become white (and your computer will beep again). Keep going until all the semilandmarks are white; the landmarks will still be red, including those at the ends of the curve. (Fig. W3.3C).

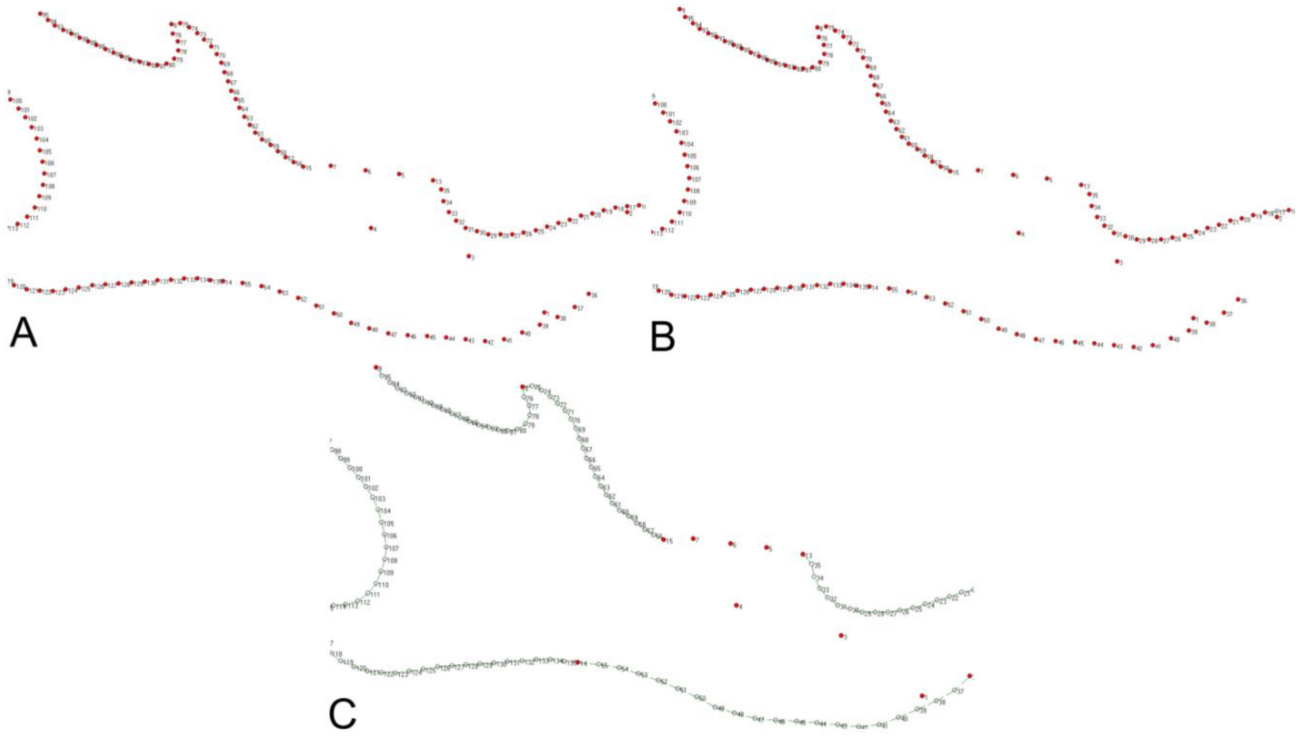


Figure W3.3. Producing the sliders file in **tpsUtil**. A: Screen shot of the program window when you first open the program; B: Screen shot after specifying the first slider; C: Screen shot after specifying all the sliders; the landmarks (that do not slide) will all be red and the sliders will all be white.

CoordGen (2D)

CoordGen offers one method for superimposing semilandmarks: the minimization of the distance between specimen and consensus. **tpsRelw** also offers the distance minimizing approach, but its default method is the minimization of bending energy of the deformation from the consensus to specimen (covered in Chapter 5 of the textbook).

The first step is to load your file. There are several kinds of datafile formats that can be loaded into **CoordGen**, including several options for various sorts of tps files. If you removed the control lines in **tpsUtil**, there are still three options that indicate whether there is a ruler or scale factor. Two of these are buttons in the orange **Load Data** box: choose the top one if you used a ruler, the second one if you have neither a ruler or scale factor. If you used the scale factor in **tpsDig**, you need to go the **File Options** menu (on the toolbar) and select **Load TPS File with Scale Factors**. If you did not remove the control lines and want **CoordGen** to do that for you, along with making your semilandmark protocol file, you will use the second or third option on the **File Options** menu. The upper one is to load a file without a ruler or scale factor, convert the curves to landmarks and create a semilandmark protocol. The one below that is for a file that has a ruler in it (it does not say that it will convert the curves to landmarks but it does). There is no option for a file that has a scaling factor, so if your file has the scale factor and you select the first option, centroid size will not be calculated properly. You can get around that problem by opening your file twice in **CoordGen**, first using **Load TPS File with Scale Factor**, which will produce a file that has centroid size calculated from the landmarks. Save the output file. Then open the file a second time using **Load tps, convert curves, create SLM proto** to convert the curves and produce a semilandmark protocol.

After loading your file, go to the **Other Tools** menu and select **SemiLandMark alignment**; **SemiLand** will open and your data will be visible in the window. Load your protocol file, then go to the **File** menu and select what you want to save, e.g, all the data (landmarks, semilandmarks plus helpers), or only landmarks plus semilandmarks, or only landmarks. You can plot the selected data using the **Replot** function. On the statistics menu there is a function to **Compute the degrees of freedom** for the configuration, but it is not yet implemented, nor is the button on the **Diagnostic Plots** for the **Tangent Estimates Plot** and **Tangent Estimates on Original Mean.** To look for outliers, you have to go back to the **CoordGen** window, and **Clear** the original data set and load the one you just created using the **Load BC File (X1Y1CS)** button in the orange box. Now go to the **Other Tools** menu and select "**Quick Diagnostic PCA**". You will see a scatter plot of the numbered specimens.

tpsRelw (2D)

To do the semilandmark superimposition in **tpsRelw**, click on **Data**, load your file, and then go to the **File** menu and select **Open sliders file**. If you want to use the distance-minimization method (not the bending energy method), go to the **Options** menu, select **Slide method** and from that menu, select **Chord-min d2**. The bending-energy method is the default, and it can be extremely slow, so do not forget this step unless you want to use the bending-energy method. Then click **Consensus**. When the consensus is computed, the semilandmark superimposition is done. To save the aligned data, go to the **File** menu, select **Save** and save both **aligned specimens** and **centroid size** files. The aligned specimens can be saved in either tps or NTS format; centroid size is saved in NTS format. To look for outliers, go through the rest of the procedure, selecting **Partial warps** and then **Relative warps**. These may both take very long to run now. When the calculations are done, go to the **Display** menu and display **Relative warps**.

Programs for landmark and semilandmark alignment (3D)

There is only one freely available program for aligning 3D semilandmarks, at present, **ChainMan3D** (IMP).

ChainMan3D

Chainman3d is an IMP6 program designed to superimpose combinations of landmarks and semilandmarks. It requires that the semilandmarks are on defined curves rather than scattered across a surface. It combines the functions of evenly spacing the digitized points, standardizing the number of points on corresponding curves, and 'sliding' using the perpendicular alignment (distance minimizing) approach. It can be used for data sets that have no landmarks other than those at the ends of curves. (If you only have landmarks, you don't need it). It can use a ruler to compute a scale factor, but one is not required. This program is designed to take data from a wide variety of 3D digitizers, so it requires you to do a more work to prepare the data, and it requires you to write a protocol file that tells it how to read your data.

Chainman does not require that there be the same number of points for each specimen, but as always, there must be the same number of actual landmarks. **Chainman** does require that the input follows a datafile format similar to that used in tps programs. The first line for each specimen should be "lm=" with the total number of points (leave blank for now). Each subsequent line should have the *x*-, *y*- and *z*-coordinates in that order for one point. As in 2D data sets, the landmarks come before curves, landmarks and curves must always be in the same order, and curves must always be digitized in the same direction (apex to base or anterior to posterior, etc.). The last line for a specimen should be “image=” followed by an identifier. Any other labeling or messages that may have been inserted by your digitizer must be removed.

To separate the curves from each other and from the landmarks that are not on curves, you will need to add a break point – a point not on the specimen that signals the end of a series (the non-curve landmarks or the curve points). While digitizing, end each series by digitizing a point far from the specimen, ideally a standardized point in the space such as its (0, 0, 0) reference or something near the limits of its reach. After digitizing, you must replace the coordinates of the break point with a standardized code such as 999 999 999. (If you did not digitize break points, you will have to count out where to insert the code.) You must include the break points (and the ruler, if present) in the count of digitized points given in the “lm=” line. If there are no landmarks or ruler before the first curve, the first line after “lm=” must be the break point code. You do not need a break point code after the last curve.

The protocol file begins with four lines describing the ruler. For example, if the first two points are the ends of a 10 cm ruler, the first four lines would be 1, 1, 2, 10 (one number on each line). The first line indicates there is a ruler, the next two give the ordinal position of its points in the data file (first and second), and the last line gives the length of the ruler (it is up to you to remember or record the units elsewhere). If there is no ruler, these lines would be 0, 0, 0, 0. The next line gives the number of curves, and the one after that gives the break code (999 in the example above). Then for each curve, there are two lines, the first giving 0 or 1 to indicate an open or closed curve, and the next giving the number of evenly spaced points to calculate from the input. Thus, the following protocol indicates no ruler, 4 curves, the break point code, and then each curve is open and has a progressively smaller number of points.

```
0
0
0
0
4
999
0
20
0
15
0
10
0
7
```

When you load the protocol file into **Chainman3d** (using the buttons in the blue-green 3D box), the information will be reported in the display box. After you load the data file, a 3D display window will open, showing the landmarks and recalculated semilandmarks (curves are not connected). As in **CoordGen**, data are blue circles, means are red symbols. If you like, you can save these data by clicking the third button. Click the last button to *both* perform the semilandmark alignment and save the resulting coordinates. This will open a second screen showing the aligned data, and a box telling you to close it to save the result. Click either **OK** or the red **X** in the corner, then save the data in the usual manner. The data will be written to an IMP-like text file, with the data for each specimen in a single line, with *x*-, *y*-, and *z*- coordinates for the first point, then the second, and so on, and centroid size in the last column.

Programs for symmetrizing data

The procedure(s) for obtaining the symmetric shapes and reducing the data to the midline landmarks plus one of the two paired landmarks (thereby eliminating the redundant coordinates) differ depending on whether (1) the data exhibit object or matching symmetry and (2) the data comprise just one individual per side or multiple replicates of each individual/slide. The first determines how the symmetric component is obtained and the second determines whether multiple replicates of each individual are averaged (across replicates and sides). There are two programs that can estimate the symmetric shapes, **MorphoJ** and **Sage**. **MorphoJ** will infer which landmarks are paired in the case of object symmetry so it does not need a file that tells it which landmarks are paired; **Sage** does. Both programs need the information on which individuals are which. In the case of object symmetry, if there is only one replicate of each individual, **MorphoJ** does not need any additional information aside from that contained in the specimen identifier in the first column of the data matrix. For matching symmetry, and for datasets that contain multiple replicates of each individual, both programs will need protocol files to identify the individuals and sides. When there are multiple replicates, the symmetric shapes are among the outputs of an analysis of fluctuating asymmetry. This analysis is covered in Chapter 12, so we do not explain what the programs are doing—we merely outline the procedure for obtaining the symmetric shape for each individual.

MorphoJ (2- and3D)

If your data have object symmetry, and you have only one replicate for each individual, i.e. each individual was photographed and digitized only once, and you have superimposed your data, you have completed the analysis. All that remains is to decide what you want to do with the coordinates of the symmetric shape. You can analyze them further within **MorphoJ**, or you can export them. Within **MorphoJ**, it is not necessary to reduce the data to the midline landmarks plus one side of the bilateral (paired) landmarks. However, if you export the data to use in another program, you will need to remove one landmark from each set of paired landmarks (either all those on the left or all those on the right). To our knowledge, **MorphoJ** does not export such a file. You can export the file that contains all the landmarks and delete one side in another program (e.g., a spreadsheet).

If you have multiple replicates of each individual, you will need to prepare a “classifier” file so that **MorphoJ** will know which specimens are which. The first column of this file must be identical to the specimen ID in your data file. If you prepared a text file with an ID in the first column, copy that column into the first column of the classifier file to ensure that the two are identical. If you load a tps file, the IDs for the shape data are taken from the ID = line in the tps file, which starts with zero, so make sure that the numerical values in the ID column of your classifer file begin with zero. The second column should be codes that identify the replicates of individuals. For example, if the first three “specimens” in the file are the coordinates for the same individual, measured three times, the first three lines of the column will all be “1”. The next three would all be “2” and so forth. Put a header in this file (e.g., “ID” for the first column, “Individual” for the second). Based on this information, **MorphoJ** can compute the average symmetric shape for each individual in the dataset. Select your superimposed data file in the **Project Tree**, file then go to the **Preliminaries** menu and select **Average observations by** and **Individual** and for the data type select **Symmetric component**. You can export those averaged symmetric shapes, but you need to be careful because **MorphoJ** has resorted your file; the first individual in your input data file is not the first in your output file. The IDs (which are now all numerical values even if you input specimen IDs that had characters in them) are sorted 1, 10, 100, 101, 102…109, 110, 111… 119, 2, 20, 201…So when you open your output file, sort on the first column to restore the order of your original file. At that point, you can replace the numerical IDs with more informative specimen IDs.

The procedure is slightly more complex for matching symmetry because the right and left sides are digitized on different images and therefore have different IDs. Even if you have just one individual measured on both sides you will need a “classifier” file to tell **MorphoJ** which IDs represent sides of the same individual. The right and left sides must have the same value on the “Individual” classifier or else **MorphoJ** cannot average the two sides. To compute the average, select your superimposed data file in the **Project Tree**, go to the **Preliminaries** menu and select **Average observations by** and **Individual**, and for the data type select **Procrustes coordinates**. You can then export this file (but see above on sorting).

Sage (2D)

Sage, written by Eladio Marquez, is not one of the programs that you downloaded already. Like the IMP programs, **Sage** was written in Matlab and compiled to run under Windows. You will therefore need to download the installer, **MCRInstaller** (http://www-personal.umich.edu/~emarquez/morph/index.html). This is not the same installer that you got when you downloaded the IMP software, but like that one, you need a username and a password. Once you get them, make a folder (e.g., Matlab7.11), download **MCRInstaller** and run it. Then download **Sage** (and the other software that interests you) and expand them in Matlab7.11/bin/win32. If you are interested in modularity and in co-evolution of traits, download **Mint** and **Coriandis**. **Mace** calculates the repeatability of matrices and matrix correlations. As you might guess from the names of the programs, this is the "Morphospice series".

To do the analysis in **Sage**, you need the data file and also files that tell **Sage** who is who, and either which landmarks on the right side go with which on the left or which side is which. The data file can be in tps format, with either ruler or scale factor (or both) or in IMP format or it can be a file of *x*-, *y*-, coordinates in rows without centroid size in the last column. The **Individual Protocol** for **Sage** is a single column. The first individual in the file is "1", the second individual is "2", etc. Every replicate has the same number in that file, meaning that all the replicates of specimen 1 have a "1" in the file. Presently, **Sage** can only analyze files that have the same number of replicates for every individual.

In the case of matching symmetry, the individuals are numbered sequentially in the file, with all the replicates for each individual together (i.e., the first replicate is in the first row, the second in the second row, the third in the third row). The **Individual Protocol** would look like Table 2 if there are five specimens, each measured twice.

Table 2. "Individual Protocol" for object symmetry in Sage. The file includes five individuals, each measured twice.

1
1
2
2
3
3
4
4
5
5

The **Pair Protocol** is for object asymmetry and it contains two columns. One column gives the number of the landmark on the left side and the other gives the corresponding landmark on the right side. The unpaired midline landmarks come first in the file; the number of the landmark is in the first column and "zero" is in the second. A fragment of a pairing protocol is shown in Table 3.

Table 3. "Pair Protocol" for object symmetry in *Sage*. Midline landmarks are listed first, with a "zero" in the second column. The paired landmarks come next, with the corresponding landmarks on the right and left sides on each line.

1	0
2	0
3	0
4	5
6	7
8	9
10	11
12	13
14	15

In the case of matching symmetry, the data should be sorted so that all the right (or left) sides come first in the file, followed by all the left (or right) sides. The individuals should be in the same order so that the same individual is first for both right and left sides, followed by the same second individual for both sides, etc. If they are not ordered that way, one individual's left side will be matched to another individual's right side. The Sides Protocol contains one column; all the right sides are coded by a "0" (zero) and all the left by "1" (it does not actually matter whether the right or left is first, so long as the side that is first is "0" and the other is "1"). The Individual Protocol is therefore also ordered so that all the right sides precede the left sides. The first individual is a "1" (just as in the case of object symmetry) and all the data for that individual and that side are given in order (i.e., the two replicates for the right side of specimen 1 are coded as "1" in the protocol file, and the data for the second replicate follows that of the first. A fragment of the Individual protocol for matching symmetry is shown in Table 4 for four individuals, each measured twice on each side.

Table 4. An Individual Protocol for matching symmetry in Sage; there are four individuals, each measured twice on each side.

1
1
2
2
3
3
4
4
1
1
2
2
3
3
4
4

To run **Sage**, you first **load** the data file (if you have a ruler in the file, specify its endpoints and scale) and click on **Proceed**. You then specify whether your data have object symmetry (like the skull) or matching symmetry (like fly wings). Click on the picture that represents your version of symmetry. After you select the type of symmetry, you need to **load** the two protocols (a pop-up window opens and says which protocol to load first, then second). If you have only one replicate per specimen and merely want to get the symmetric shape, do not run any analyses, just click on **Symmetrize data** (at the bottom of the interface). A pop-up window will open so you can name your output file, i.e., the one into which the coordinates will be saved. If you have multiple replicates, however, you want to produce the averaged symmetric shape for each individual. In that case, you need to run the analysis. Unless you are interested in the statistical analysis of fluctuating asymmetry, you do not need to worry now about the assumptions of the test. So do only the **Procrustes Anova** and make sure that you are not asking for any permutations, and make sure that the **Manova** option is not selected—these analyses take a long time to run. The **Procrustes Anova** without permutations is very fast, so run that. To get the coordinates for your symmetric shapes, go to the **File** menu. First you should select your save options. Go to the **Save options** menu; the default for matching symmetry is to save the midline coordinates plus the coordinates for one half of the paired landmarks, i.e., those for one side. If that is your preference, you needn't do anything. Otherwise, select the option that you want (you can get the whole configuration as well as half if you want to do analyses with the half configurations but show the pictures on the whole). Go to the **Save fitted data** menu and select **Individual component (symmetry)**. The output file will be in IMP format.

Program for estimating coordinates of missing landmarks

OSymm() (2- and 3D)

OSymm(), written by Annat Haber, is the only program that we know of that uses object symmetry to estimate the coordinates of missing landmarks. **OSymm()** reads both 2- and 3D data. The input file is in *k* x *m* x *N* format, meaning that there are *k* landmarks, with *m* dimensions for *N* specimens. The datafile format, however, is not precisely the **shapes** format because the first row contains the IDs for the specimens (above the *x*-coordinate for the first landmark for each specimen) and the first column contains the names of the landmarks. Table 5 (on page 25 and 26) shows an example data file (provided by Annat Haber). Missing landmarks are coded as NA in the data matrix. The first line will be skipped when the data are read into the data matrix. The first *column* is *not* skipped, and it is important for two reasons. First, the landmark names in that column will be used to tell **Osymm()** which landmarks are along the midline or on the right or left sides. Second, the landmark names will be written to the output file so you can figure out which coordinates are which (they will be sorted in the output file, with midline landmarks coming first, followed by right then left). The names make it easier to figure out the output.

Table 5. Example data file.

	RMNH-17379			RMNH-20100			RMNH-2064		
tPM	160.94	-389.49	59.23	221.84	-395.29	59.96	148.10	-404.15	59.48
IOF	101.77	-406.32	47.79	166.18	-414.91	49.34	90.46	-422.31	46.40
P3	99.07	-411.41	33.09	163.53	-421.16	36.65	87.15	-426.41	31.79
abM1	84.16	-419.17	25.60	149.70	-427.87	28.57	70.78	-432.18	23.44
adJS	31.83	-433.49	45.09	97.94	-442.61	46.45	18.21	-446.69	43.31
vZA	17.96	-434.92	42.38	83.20	-444.46	42.17	2.33	-446.99	40.89
PSO	-30.83	-405.73	47.74	37.68	-422.58	48.93	-42.24	-420.16	47.20
PS(as)	NA	NA	NA	90.16	-420.93	47.06	11.36	-425.39	45.10
JFO	36.64	-436.93	57.08	102.05	-443.36	56.83	23.61	-450.46	54.28
LFO	57.21	-417.21	71.74	123.94	-425.72	72.47	46.74	-432.89	71.65
JLO	63.24	-424.17	62.11	128.34	-432.25	61.81	52.30	-439.19	57.96
aLJ	73.86	-411.73	59.98	142.98	-421.39	58.36	65.65	-427.60	57.05
pMN	73.89	-399.95	75.37	145.27	-407.84	74.10	66.78	-414.91	73.02
SNa	115.51	-393.69	75.67	180.97	-401.04	72.60	104.11	-410.03	74.87
aN	116.66	-387.92	79.38	181.52	-393.81	76.37	107.07	-401.89	78.67
FFN	59.03	-389.39	78.71	126.28	-397.20	78.59	54.32	-402.85	79.08
FFP	-1.89	-391.70	81.52	71.69	-402.09	82.45	-13.30	-404.35	84.29
SOP	-39.65	-393.61	37.54	27.96	-409.04	38.78	-50.25	-404.84	42.75
tPM_L	160.51	-383.19	59.71	221.97	-389.99	60.39	148.47	-398.19	59.97
IOF_L	99.83	-370.59	47.56	163.32	-378.25	47.06	89.71	-380.60	48.11
P3_L	96.66	-365.85	33.33	160.46	-375.11	33.31	85.75	-376.13	33.66
abM1_L	80.76	-359.08	24.86	145.50	-370.83	23.96	68.70	-370.50	25.61
adJS_L	28.65	-349.03	44.25	NA	NA	NA	16.47	-359.52	43.26
vZA_L	15.94	-348.13	41.95	76.04	-364.42	36.22	0.90	-360.75	41.05
PSO_L	-30.99	-378.93	47.05	36.36	-391.17	47.77	-42.37	-388.60	46.91
PS(as)_L	20.76	-374.12	42.92	87.77	-384.53	44.46	10.34	-381.63	45.52
JFO_L	32.69	-344.52	56.41	96.24	-360.18	48.52	21.38	-355.82	54.74
LFO_L	54.69	-361.90	71.29	118.92	-372.26	69.19	45.56	-373.13	72.31

Table 5. Cont’d.

	RMNH-17379			RMNH-20100			RMNH-2064		
JLO_L	59.91	-355.62	61.87	122.22	-365.28	54.92	50.85	-365.20	59.05
aLJ_L	73.13	-367.11	59.86	140.31	-373.87	53.47	64.31	-377.27	57.47
pMN_L	73.76	-378.45	75.23	143.17	-385.32	73.22	65.96	-390.17	73.76
SNa_L	115.29	-381.01	75.73	180.92	-386.32	71.58	103.87	-393.44	75.97
aN_L	117.27	-386.86	79.23	182.13	-392.58	76.38	106.64	-401.24	79.00
V1	113.67	-408.56	41.28	60.77	-403.01	37.16	108.81	-393.40	45.04
V3	78.10	-407.41	97.77	11.94	-403.55	75.48	74.58	-393.22	101.32
BMF	25.96	-406.92	145.24	-47.34	-405.37	100.42	21.11	-390.46	146.60
OMF	9.32	-406.23	147.74	-65.29	-405.50	93.49	3.91	-388.72	147.69
SOP_V	-10.69	-407.73	138.96	-81.34	-406.17	80.45	-16.98	-385.37	135.71
BOC	23.19	-396.05	140.07	-49.51	-395.81	94.31	19.60	-379.26	142.45
F8	33.19	-394.18	120.08	-36.24	-391.78	78.16	29.40	-375.35	122.68
PS(as)_V	36.29	-390.36	99.74	-23.88	-387.48	59.64	34.93	-366.96	101.29
JFO_V	40.43	-362.24	80.24	-16.68	-363.38	48.72	42.02	-341.26	88.34
JLO_V	60.36	-374.38	61.90	8.82	-371.69	37.90	63.06	-353.09	69.18
pJM	55.39	-372.83	95.65	-5.29	-371.17	65.25	59.71	-355.76	99.51
pbM3	72.48	-381.82	100.13	10.11	-375.61	76.47	73.91	-363.46	104.74
P3_V	106.77	-385.81	66.30	NA	NA	NA	105.49	-369.66	70.95
P2dia	108.89	-393.67	57.48	55.27	-387.90	48.82	106.14	-377.23	60.92
BOC_L	24.66	-416.31	140.32	-47.59	-413.94	94.99	17.85	-399.64	140.60
F8_L	31.92	-419.71	121.81	-34.85	-418.69	79.96	NA	NA	NA
PS(as)_LV	37.27	-424.99	101.22	-22.13	-424.04	61.51	30.08	-410.33	98.44
JFO_LV	40.95	-454.70	83.61	-12.22	-447.06	54.77	32.20	-435.44	82.93
JLO_LV	60.29	-443.12	64.10	11.26	-439.03	43.08	55.31	-426.60	63.94
pJM_L	55.15	-443.41	98.58	-1.41	-437.61	68.09	50.89	-426.21	95.38
pbM3_L	72.88	-433.02	102.73	12.86	-430.70	79.24	66.93	-422.85	101.14
P3_LV	106.35	-431.09	67.69	52.18	-425.55	56.89	99.15	-419.33	66.50
P2dia_L	108.29	-422.86	58.78	55.84	-417.35	50.47	101.58	-410.83	58.12

To run **OSymm()**, you need to read in the data and tell **R** which landmarks are along the midline, or on the right and left sides. Here, we read in a datafile that we will call XX; this file contains the specimen IDs in the first row. The first step is to produce the matrix, containing the coordinates for a single specimen, using the "`as.matrix`" statement. The landmarks are 3D so if we want coordinates for the third specimen we need columns 7 through 9. This is written as [,7:9]; the square brackets surround indices; the first index is for rows, the second for columns. We want all rows, so we leave the row index blank; after the comma, we specify the columns we want, written as 7:9. Next, we tell **OSymm()** which landmarks are “midline”, “right” and “left”. To do that, we define three vectors, “midline” “right” and “left” by concatenating the coordinates of the named landmarks. (Concatenate means to link in a series, so we are making a vector by appending the data for all the named coordinates in a series, ordered according to the order of their names). Fortunately, you only have to write this out once.

```
XX <- read.table(file.choose(), row.names=1, skip=1)
X <- as.matrix(XX[,7:9])
midline <- c("FFN", "FFP", "SOP", "V1", "V3", "BMF", "OMF", "SOP_V")
right <- c("tPM", "IOF", "P3", "abM1", "adJS", "vZA", "PSO", "PS(as)",
"JFO", "LFO", "JLO", "aLJ", "pMN", "SNa", "aN", "BOC", "F8", "PS(as)_V",
"JFO_V", "JLO_V", "pJM", "pbM3", "P3_V", "P2dia")
left <- c("tPM_L", "IOF_L", "P3_L", "abM1_L", "adJS_L", "vZA_L", "PSO_L",
"PS(as)_L", "JFO_L", "LFO_L", "JLO_L", "aLJ_L", "pMN_L", "SNa_L", "aN_L",
"BOC_L", "F8_L", "PS(as)_LV", "JFO_LV", "JLO_LV", "pJM_L", "pbM3_L",
"P3_LV", "P2dia_L")
```

On the first run of **OSymm()**, we impute the missing landmark coordinates for this one specimen. First, if you have not already done so, read the function **OSymm()** into **R**:

```
OSymm <- function(X, midline, right, left) {
      ncl <- ncol(X)
      Xr <- cbind(X[,-ncl], -X[,ncl])
      Xow <- Xo <- rbind(X[c(midline, right, left),])
      Xrw <- Xr <- rbind(Xr[c(midline, left, right),])
      rownames(Xrw) <- rownames(Xr) <- rownames(X)
      Xo[which(is.na(Xr))] <- NA
      Xr[which(is.na(Xo))] <- NA
      mo <- matrix(apply(Xo, 2, mean, na.rm=TRUE), byrow=TRUE, nr=nrow(Xow),
      nc=ncol(Xow))
      mr <- matrix(apply(Xr, 2, mean, na.rm=TRUE), byrow=TRUE, nr=nrow(Xrw),
      nc=ncol(Xrw))
      Xrwc <- Xrw-mr
      SVD <- svd(t(na.omit(Xr-mr)) %*% na.omit(Xo-mo))
      L <- diag(SVD$d)
      S <- ifelse(L<0, -1, L)
      S <- ifelse(L>0, 1, L)
      RM <- SVD$v %*% S %*% t(SVD$u)
      Xrot <-(Xow-mo)%*% RM
      SC <- apply(array(c(Xrwc,Xrot), dim=c(nrow(Xrot),ncol(Xrot),2),
      dimnames=list(rownames(Xrot),colnames(Xrot))), 1:2, mean, na.rm=TRUE)
      Xrot[which(is.na(Xrot))] <- Xrwc[which(is.na(Xrot))]
      Xrwc[which(is.na(Xrwc))] <- Xrot[which(is.na(Xrwc))]
      list(rec.orig=Xrot, symmconf=SC, rec.ref=Xrwc)}
```

Then we run it on our data; at the command prompt, type:

```
FixThird <- OSymm(X,midline,right,left)
```

There are three values in **FixThird** (1) $rec.orig, (2) $symmconf, and (3) $rec.ref. The values that we want are for the symmetric configuration (**FixThird$symmconf**).

To fix the second specimen, we first need to put its data in the matrix **X**; so we type

```
X <- as.matrix(XX[,4:6])
```

This overwrites the matrix **X** that we produced for the third specimen (if you do not want to do that, just call the first one **X3**, this one **X2**, and the next **X1**). Then we rerun **OSymm()**,

```
FixSecond=  OSymm(X,midline,right,left)
```

And we do the same for the first (after creating the matrix with the data from columns 1:3)

We do not need to rerun the function for each individual—we could write a loop that goes through the entire file, but we will save loops for later.

After running the function for all specimens, we want to save the coordinates either in a ***.RData** file or to a text file. But you might not want to have a separate file for each specimen; so before exporting the file we can combine the output into a single file. We will use "**cbind**" which merges files by columns.

```
FixedData <- cbind(FixFirst,FixSecond,FixThird)
```

Then we will save it to a comma-delimited (**.csv**) file

```
write.csv(FixedData,file.choose()).
```

Because you used "**file.choose())"** a window will open (taking you to your default folder) and you can name the file whatever you want. The file that you name does not already exist so you will get a query asking if it should be created. There is no option to reply "Of course" so just answer "Yes".

Literature Cited

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

4

Theory of Shape

Problems and exercises on the theory of shape

To get a feel for what the software will be doing for you, do these problems and exercises using pencil, paper and a scientific calculator. The numbers of landmarks are small, to keep the level of tedium to a minimum!

1. Suppose that the configuration matrix for a given shape is:
    a. How many landmarks are there in this configuration? How many dimensions does it have?

$$\mathbf{A} = \begin{bmatrix} 0.0 & -1.0 \\ 0.0 & 0.5 \\ 0.7 & -0.2 \end{bmatrix}$$

b. Sketch the shape representing this configuration (you may want to use graph paper, if it helps). Number the landmarks.
c. Write out the row vector form of this landmark configuration.
d. Find the centroid position of this landmark configuration. How many coordinates are in the centroid position? Sketch the location of the centroid on your picture from (b) above.
e. Write out the centered form of this configuration matrix, by subtracting the value of the *X*-coordinate of the centroid from each of the values in the first column, and subtracting the value of the *Y*-coordinate of the centroid from each of the values in the second column.
f. Find the centroid size of this landmark configuration.
g. Now form the pre-shape configuration for **A**. Do this by dividing the centered form

of this matrix (solution to (e) above) by the centroid size (solution to (f) above). Remember that when you divide a matrix by a scalar (an ordinary number, like centroid size), you must divide each value in the matrix by the scalar divisor (which is centroid size in this case).

2. Suppose a configuration of dimensional landmarks is given by:

$$B = \{0.3, -1.0, 0.25, -0.4, 0.0, 0.75, -0.2, 0.35\}$$

   a. How many landmarks are in this configuration?
   b. Write out the configuration matrix for this configuration.
   c. Find the centroid for this configuration.
   d. Find the centroid size for this configuration.

3. Given the landmark configuration:

$$C = \{0.1, 0.1, 0.1, 0.3, -1.0, 1.1, -0.6, -0.3, 0.2, 0.3, -0.1, 0.15\}$$

   a. Can you determine what K and M are?

4. Suppose we have the configuration matrix:

$$\mathbf{X} = \begin{bmatrix} 0.5 & 0.5 \\ -0.2 & 0.3 \\ 0.1 & 0.3 \end{bmatrix}$$

   a. Compute a configuration matrix that would represent **X** in pre-shape space.
   b. Now, for the truly stout of heart, suppose we have a second configuration matrix in pre-shape space:

$$\mathbf{Y} = \begin{bmatrix} 0.6864 & 0.1961 \\ -0.6824 & -0.0981 \\ 0.0 & -0.0981 \end{bmatrix}$$

Determine the angle that you would have to rotate the pre-shape space matrix form of **X** (from (a) above) to produce a partial Procrustes superposition of **X** on the reference form **Y**.

5. Given two matrices:

$$\mathbf{X} = \begin{bmatrix} 0.7146 & 0.2150 \\ -0.6438 & -0.0913 \\ -0.0709 & -0.1237 \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} 0.6864 & 0.1961 \\ -0.6824 & -0.0981 \\ 0.0 & -0.0981 \end{bmatrix}$$

a. where **X** is in partial Procrustes superposition with **Y**:
b. Find the partial Procrustes distance between the two.
c. Use the partial Procrustes distance to find the Procrustes distance between the two.
d. Use the Procrustes distance to calculate the full Procrustes distance between the two.

Answers to problems and exercises

(A full solution is given if the calculation has not been seen before.)

1. Looking at the configuration matrix:
   a. There are three landmarks (*K* = 3 rows) and each is in two dimensions (*M* = 2 columns).
   b. See Figure 4.19.
   c. In row form, **A** = {0.0, –1.0, 0.0, 0.5, 0.7, –0.2}.
   d. The centroid is located at (0.2333, –0.2333), or *X* = 0.2333, *Y* = –0.233.

The centroid position is calculated:

$$X_C = \frac{(0+0+0.7)}{3} = 0.2333$$
$$Y_C = \frac{(-1+0.5-0.2)}{3} = -0.2333$$

Figure 4.20 shows the location of the centroid.

   e. We simply subtract the *X*-coordinate of the centroid (0.2333) from the first column of **A**, and subtract the *Y*-coordinate of the centroid (–0.2333) from the second column. This leaves us with:

$$\begin{bmatrix} -0.2333 & -0.7667 \\ -0.2333 & 0.7333 \\ 0.4667 & 0.0333 \end{bmatrix}$$

Note that if you add up the values in the first column you get zero, which is also true for the second column. Thus the centroid position of the centered matrix is (0, 0).

f. The centroid size is 1.2055. The centroid size is the square root of the summed squared distances of the landmarks from the centroid, which is:

$$CS = \{(0 - 0.2333)^2 + (-1 - (-0.2333))^2 + (0 - 0.2333)^2$$
$$+ (0.5 - (-0.2333))^2 + (0.7 - 0.2333)^2$$
$$+ (-0.2 - (-0.2333))^2\}^{1/2}$$
$$= 1.2055$$

An easier approach is to use the centered form of the configuration matrix (with the centroid set to zero). With this form, we can take the square root of the summed squared coordinates of the landmarks:

$$CS = \{(-0.2333)^2 + (-0.7667)^2 + (-0.2333)^2$$
$$+ (0.7333)^2 + (0.4667)^2 + (0.0333)^2\}^{1/2}$$
$$= 1.2055$$

g. The resulting pre-shape space configuration is

$$\mathrm{A}_{\text{pre-shape}} = \begin{bmatrix} -0.1936 & -0.0630 \\ -0.1936 & 0.6083 \\ 0.3871 & 0.0277 \end{bmatrix}$$

Note that the entries are identical to the centered matrix values (see (e) above) divided by 1.2055. 2. Looking at the configuration:

2. a. There are four landmarks.
   b. The configuration matrix is:

$$\begin{bmatrix} 0.3 & -1.0 \\ 0.25 & -0.4 \\ 0.0 & 0.75 \\ -0.2 & 0.35 \end{bmatrix}$$

   c. The centroid is located at $X_C = 0.0875$, $Y_C = -0.075$.
   d. The centroid size $CS = 1.4087$.
3. No! This might be $K = 6$ and $M = 2$ (a two-dimensional system), or $K = 4$, $M = 3$ (a three-dimensional system). If the data are in a row format, you cannot tell the value of K or M from looking at it.
4. Looking at the configuration matrix:
   a. The pre-shape space form of **X** is

$$\begin{bmatrix} 0.7013 & 0.2550 \\ -0.6376 & -0.1275 \\ -0.0638 & -0.1275 \end{bmatrix}$$

b. The triangles can be iteratively rotated, or Equation 4.13 can be used:

$$\theta = \text{arctangent}\left(\frac{\sum_{j=1}^{K} Y_{Rj}X_{Tj} - X_{Rj}Y_{Tj}}{\sum_{j=1}^{K} X_{Rj}X_{Tj} + Y_{Rj}Y_{Tj}}\right) \qquad \textbf{(4.13)}$$

Substituting the appropriate values of *X* and *Y* for the reference (R) and target (T) yields:

$\theta = \tan^{-1}\{(0.1961 \times 0.7013 + (-0.0981) \times (-0.6376) + (-0.0981)$
$\times (-0.0638) + (-0.6864) \times 0.2550 + (-0.6864) \times (-0.1275)$
$+ 0 \times (-0.1275))/(0.6864 \times 0.7013 + (-0.6864) \times (-0.6376)$
$+ 0 \times (0.0638) + 0.1961 \times 0.2550 + (-0.0981) \times (-0.1275)$
$+ (-0.0981) \times (-0.1275))\}$

$\theta = -0.0562$ radians $= -3.2175°$

5. Looking at the matrices:

a. To find the partial Procrustes distance between the two, we take the square root of the summed squared differences in the landmark coordinates:

$Dp = \{(0.7146 - 0.6864)^2 + (0.2150 - 0.1962)^2$
$+ (-0.6438 - (-0.6864))^2 + (-0.0913 - (-0.0981))^2$
$+ (-0.0709 - 0)^2 + (-0.1237 - (-0.0981))^2\}^{1/2}$

$= 0.0933$

b. Because $\rho = 2 \arcsin(D_p)$, $\rho = 2 \arcsin(0.0933/2) = 0.0933$ radians; the two are equal through three decimal places.

c. $D_F = \sin(\rho)$, so $D_F = \sin(0.0933) = 0.0932$.

5

Visualizing Shape Change

One of the principal advantages of geometric morphometrics is that the mathematical descriptions of shape differences can be rendered as graphical images depicting where the shapes differ, in what direction, and by how much. Those differences might be shown as plots of the superimposed shapes or as pictures of shape transformation (deformed grids or vectors of landmark displacement). These shape differences may reflect an axis of sample variation, or a correlation with another variable (e.g. size, age, elevation, diet); so the analytic programs will also produce scatterplots showing the distribution of individual shapes (single specimens, means of groups or taxa) along those axes or trends. Naturally, all of the packaged programs have numerous options for tailoring those graphics, some that control content and some that control esthetics. In this chapter, we describe the options for controlling the various types of graphs these programs can produce.

Within a package, many programs produce three main kinds of graph: a superimposition graph, a shape difference graph, and a scatterplot. The different programs in a package that produce a particular type of graph (e.g. all the IMP programs that show superimpositions) tend to have many shared options for displaying a particular feature (such as the same symbols and colors). However, there are also programs in each package that have other outputs tailored to more specialized applications. In this chapter, we focus on the programs you have already encountered, reserving specialized outputs for the few chapters that might call for them. In packages that analyze both 2D and 3D data, we first describe all of the 2D programs, then we describe the 3D programs by focusing on the features that differ from their 2D counterparts. We also describe how to transfer graphical outputs to commonly used graphics and presentation software. As mentioned previously, the morphometrics programs are primarily written for machines running Windows; consequently, the instructions (and commentary) are mainly written with that platform in mind.

Image file formats

Before getting into the specifics of the morphometric programs, we briefly discuss the problem of turning graphs and drawings into images on screens and printers. The fundamental problem is that the image produced on the screen or printer must take the forms of rows of cells ("dots" or pixels; voxels for 3D volumes), but the shapes that we would draw have angles and curves that don't exactly fit the grid, and colors or shading that change in gradients rather than discrete steps.

One solution is to make the grid as large and dense as possible so the difference between one dot and the next is too small to perceive. The more information you pack into each pixel the better this works. This is the approach taken by TIFF (Tagged Image File Format, *.tif) and it explains why these files are so large – they are maximizing information storage. TIFFs are not very efficient, they take a lot of disk space to store and a lot of time to load and, if you display them at a small size, you can't actually see most of that information. Consequently, compression (coding) routines were devised to reduce space. JPEG (Joint Photographic Experts Group, *.jpg) is designed to optimize file size relative to visual appearance, so simple images produce smaller files than complex ones; it also means that information is lost in the compression, which is why enlarged presentations of small jpegs look poor. PNG (Portable Network Graphics, *.png) is designed for graphics, rather than photographs, so it tends to produce high quality images of graphs with smaller files than TIFF, and less loss than JPEG. (There also are compression routines that can be applied within the framework of the TIFF format.) All these coding routines preserve the fundamental grid structure.

An alternative approach is vector graphics, designed to represent objects as much as possible as formulae for vectors and shapes. Graphic data presentations like bar charts and scatterplots, and even rather complex curves can be very effectively and economically stored in these formats. This approach is the basis of PostScript (*.ps, *.eps) and SVG (Scalable Vector Graphics, *.svg). However, photograph-like images, with complex shapes and soft edges and gradual transitions in color usually cannot be effectively compressed using such algorithms. A compromise is to use vector graphics where possible and bitmap image coding when necessary, as in Window metafiles (*.wmf, *.emf) and later generations of PostScript format (*.eps). One of the formats that tends to work well for all types of images and that can also be edited is PDF.

Most morphometrics programs will offer several options for saving images. Unfortunately, many of the coding formats have gone through several generations with different degrees of upward and downward compatibility and, more disconcerting, there can be many different variants of the same basic algorithm producing incompatibilities between programs supposedly using the same format (e.g. a program reading TIFF formats may not be able to read all the variant TIFF-compatible compression routines). A type of image file that works on one machine might not work so well on another, and a type of image file that works one week might not work so well the next. Consequently, we cannot make blanket recommendations, only general suggestions concerning strategy. Our main suggestion is to try everything. For example, if you want a regression, run the analysis in all the programs that do regressions, save each graph in all of the available formats, then insert each output into all of the various presentation and graphics programs you might be using and evaluate the quality of the results. When comparing the various methods of exporting graphical results, one thing you should consider is whether you intend to edit the graph or use it "as is". If you don't want to edit the graph, just copy and insert it into a document or presentation, consider copying it to the clipboard in the programs offering that option (but watch for unexpected cropping near edges and changes in length-width ratio). If you want to save the graph for inserting into documents later, or distributing to colleagues, choose an uncompressed TIFF option if you don't care how big the file will be or PNG if you do. If you do want to edit the graph, you may have to experiment to figure out what works with your graphics program. For example, Serif DrawPlus X2 on at least some Windows systems will allow you to paste from the clipboard and ungroup the elements for editing, but if you had saved the image (in any format) and insert that picture into a new drawing, the elements cannot be ungrouped. Using Adobe Illustrator CS2 on the same Windows systems, it is possible to ungroup and edit a graphic saved as *.eps or *.svg, but not one saved in other formats, nor is it possible to paste from the clipboard. And, as always, your result will depend on the compatibility of the version of encoding used to save the file and the version of the program used to open it.

An additional note concerning Windows vector graphics: there can be distortions of aspect ratio (height-width ratio) that depend on the versions of software used. Unfortunately, there is no simple way to anticipate this problem. If you encounter it when copying to the clipboard from an IMP program, look for a button or menu item that mentions the image aspect (ratio). Otherwise, your options are to try to correct the distortion or save it to a file in a different format.

Wireframes

The first time you load a set of 3D coordinates, the resulting plot probably is not very interpretable even if you are familiar with the source of the data. The purpose of the wireframe is to help you interpret what you are seeing in the graph.

A wireframe is a set of lines (wires) connecting the digitized points on a shape. The wires need not connect each point to some other; they would certainly not be helpful if they connected every point to every other. They are often most useful when they follow major anatomical features of the shape – ridges and edges. Both **MorphoJ** and the **IMP** programs for 3D data allow you to load a text file that tells the program which points to connect with wires. Depending on the application, the wireframe is drawn on the sample mean or other reference shape. A second wireframe may then be drawn on the target shape that is the result of the deformation.

The wireframe is strictly an aid for visualizing shapes. It is not needed to run any analysis. It also is not a guide to inferring the deformations of the space between landmarks; except in the special case of the "soft" wireframes in **MorphoJ**, the wireframes connecting the landmarks are always drawn as straight lines. The vertices of the wireframe show where the landmarks of the two shapes differ in location. Unless the deformation is uniform, the line between the landmarks will not be straight after the deformation. However, if you select the soft wireframe in **MorphoJ**, the wires in the target shape are drawn as curves reflecting the deformation of the space between landmarks as inferred from the thin-plate spline.

IMP Programs

CoordGen (2D)

After you have loaded the data (and entered the information for scaling and baseline, if necessary), the superimposed shapes are written to the graphics screen, with blue circles for every landmark of every individual and a red symbol for the mean of each landmark. The **Numbers on Landmarks** button (in the purple box near the Exit button) will put numbers near the mean of each landmark, and a tick mark on the mean pointing toward the number. This mark may be difficult to see if your data are very dense. At the bottom of the window is a box allowing you to display single shapes. Enter a number in the small box and click the **Plot Specimen** button. The landmarks for that specimen will appear on the graphics screen (with tick marks and numbers if that button is still on).

Figure Options Menu

Most choices in this menu have self-explanatory labels. **Clear Axis** removes the white box behind the graph as well as the axes and their scales. The symbol size options will not affect the thickness of the lines used to draw the symbols or the size of the numbers written next to the landmarks (if that option is on). If your life is terribly deficient in entertainment value, there is an option to replace the cursor with "Spiffy Fish". The **Zoom In** button activates the zoom function and replaces the cursor with a magnifying lens. If you now click on the screen, it will zoom in by pre-set steps; you can also click and drag the box to select an area to be enlarged to fit the plot box. There is no zoom out button, but you can undo the zoom by going back to the menu and selecting **Restore original size**. Note that this button does not turn off the zoom function, at present, the only method of doing that is to exit and restart the program. Zooming and most other alterations to the graph will revert to their defaults when you click one of the display buttons. The exception is that the axes will remain cleared.

Figure W5.1A shows the squirrel jaw data used in other demonstrations (*weslm.txt*) saved from **CoordGen** as an uncompressed *.tif. The image was made using the default settings in **Coordgen**, so the

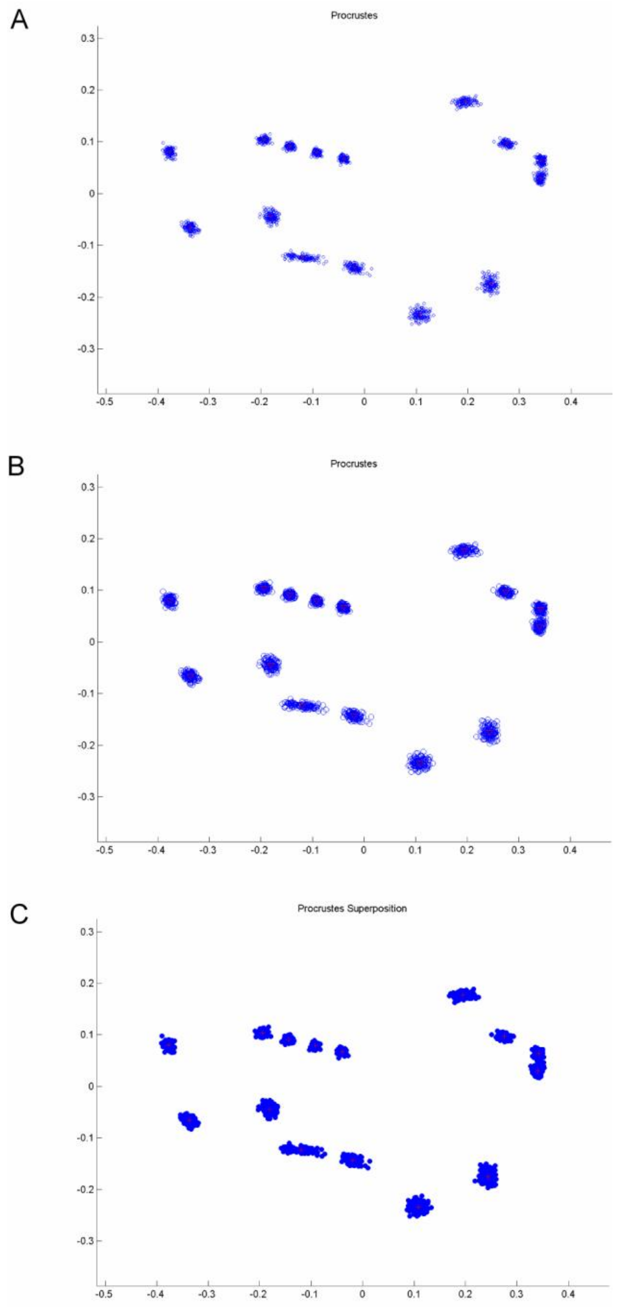


Figure W5.1. Images of superimposed shapes from **CoordGen** illustrating options to enhance readability. A: Default settings. B: Increased symbol size. C: Filled and enlarged symbols. All images were saved as uncompressed tif.

symbols are small, and drawn with thin lines. The symbols and the numbers on the axes also look a bit fuzzy, which is a reflection of the fact that this is a bitmap. If you replicate this image and enlarge it in PhotoViewer or something similar you will see the circles are made of many small dots. Images saved using as *.png or compressed *.tif are not noticeably different; the image saved as *.jpg has visible noise in the white space between symbols. The *.eps format was read as black and white by Adobe products. Images could not be produced using *svg format. Figure W5.1B shows the same data with the symbols increased a few times, which can make the symbols easier to see, although when symbols are close together, they will overlap more. In Figure W5.1C, the symbols were filled, which helps the clusters stand out when the image is reduced for printing but also makes it harder to see overlapping points in large data sets.

Options for revising and editing the quick PCA graph are not discussed here; better graphs for distribution can be obtained from other programs. Graphics options for **Semiland7** (the semilandmark alignment tool) are the same as for the main program.

Export options

CoordGen7a has two bright yellow buttons for copying to the clipboard; one (Uneven Aspect) should help address changes in aspect ratio that can occur when copying to the clipboard. You can also go to the **Figure Options** menu to select **Copy Image to Clipboard**. To save the image to a file, click the bright yellow **Print Image to File** button near the bottom of the screen (the gray one does not work), and choose from the available options. Whichever option you choose, the graphic screen will enlarge greatly. This is normal behavior for most IMP programs. Relax, nothing is broken. If you choose to copy to the clipboard, the image will briefly expand, then contract. If you choose **Print Image to File**, the image will stay expanded until you choose a format. After you complete your selections and the writing is finished, the screen will return to normal size. However, if you cancel the operation, the graphics screen will stay enlarged. You may be able to restore the graphics screen to normal size by maximizing the **CoordGen** window, or you may have to **Exit** and restart the program.

PCAGen7a

This is one of three programs for analyzing 2D data that has numerous options for manipulating the graphs they produce, the other two are **TwoGroup** and **CVAGen**. These three produce the same basic types of graphs (superimposed shapes, scatterplots of scores and shape differences) and there are few differences between programs for a particular type of graph. The other programs tend to have more limited graphical displays, which are primarily a reduced set of the options available in these three. Accordingly, we focus on the options for one program, **PCAGen**.

Superimposed shapes and the Group List

After your coordinate data are loaded into **PCAGen**, you will have the option to use different symbols to label groups in the data. If your data come from a single homogeneous population (or if you do not wish to plot by groups) you can select the **No group list** option. You will then see the superimposed shapes as black circles. Go to the **Display Options** menu and select **Show Symbol Codes** at the bottom of the list. A set of 24 colored symbols and numerical codes will appear on the screen. In **PCAGen**, and some other programs, a list of these numbers as a single column in a text file (called a Group List) can be use to indicate what symbols should be used on the plot. In wes *groups.txt*, the group list for *weslm.txt*, there are 69 "1"s, 23 "3"s and 27 "22"s; these groupings represent the geographic locations where the specimens were collected. Load *weslm.txt*, select **Load Group Membership List**, then load *wes groups.txt*. The plot will now have solid, colored symbols: black circles, red stars and blue squares (Figure W5.2). It should now be apparent from the distributions of those symbols that the groups have different mean shapes.

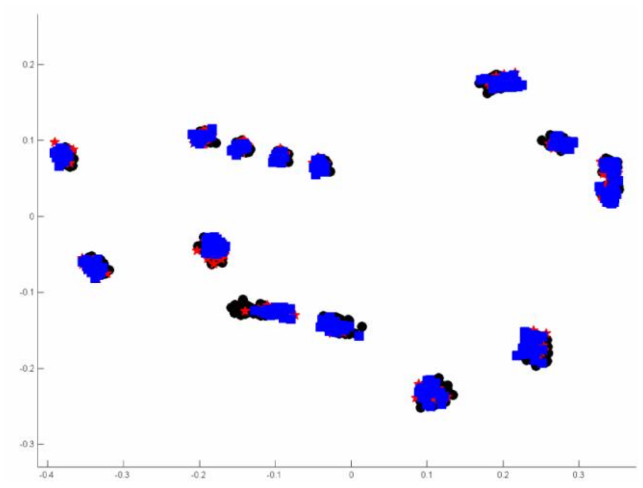


Figure W5.2. Images of superimposed shapes from **PCAGen**, coded to distinguish geographic populations by symbol color and shape.

Options to modify this graph are mostly the same as in **Coordgen**, but they are now split between two menus. Remove and restore axes and zoom on or off and restore original plot size are now under **Axis Controls**. The **Display Options** menu has the options to use filled or empty symbols and to change symbol sizes; all of these options revert to their defaults when the plot is redisplayed. At the top of the menu is an option to increase line weights, which can be used to draw thicker lines around the symbol (most effective when empty), but it does not alter the line weight for the axes or the font size of the scale. This menu also has an option to use black and white symbols (useful for publication) but not all symbols can be converted and some may be converted to a different shape. You will have to redisplay the plot for this option to take effect (here, using one of the buttons in the green **Show Landmarks** box). To revert to color symbols, go to the same place on the **Display Options** menu; the item is now named **Return to Default Symbols**. The **Display Options** menu also has several options that are not applicable to this type of plot; we will describe them when we demonstrate those plots.

Scatterplot of shape variation

Most IMP programs will produce a plot of scores on axes of variation or difference. In **PCAGen**, that plot is produced by clicking **Show PCA Plot** at the top of the purple box (Chapter 6 discusses what this plot means and the other buttons in this box). The options to change this plot are the same as for the superimposed shapes and work in the same way. In scatterplots like these, increasing symbol size can have a dramatic effect. Figure W5.3A shows the default symbol sizes – much smaller than they were on the screen; Figure W5.3B shows the result after clicking **+3 Symbol Size** twice.

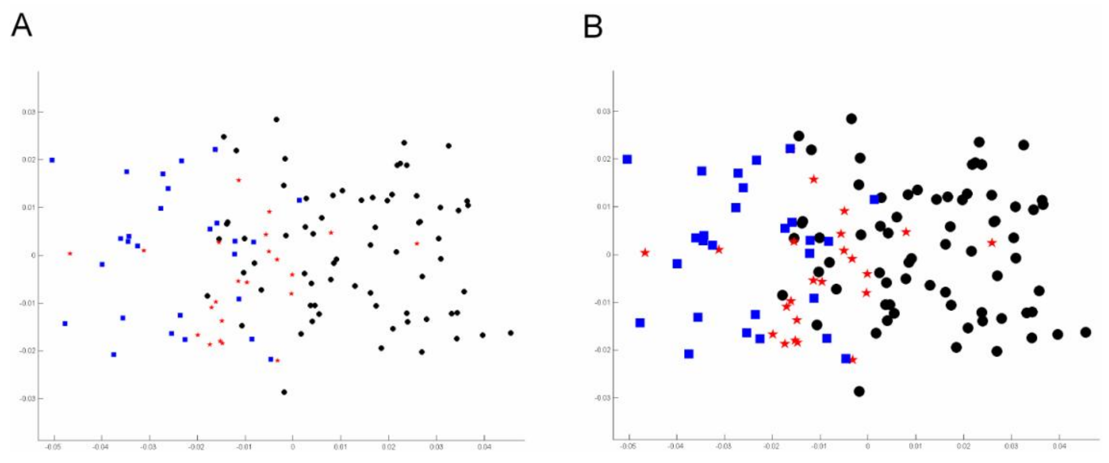


Figure W5.3. Scatterplot of PC scores produced by **PCAGen**, coded to distinguish geographic populations. A: Default symbol sizes. B: Symbols enlarged by clicking **+3 Symbol Size** two times.

Displaying shape differences as deformations

PCs are axes of shape variation and **PCAGen** has two sets of options (in the silver and brown boxes) to display what those axes mean as pictures of shape deformation. For now, we will just use one of the silver buttons, in the next chapter we describe how to generate graphs with the buttons in the brown box; the resulting deformation images can be manipulated in the same ways as we describe below.

Click the silver **Show Def. (Procrustes)** button to request a deformation picture. The default deformation image is a deformed grid with open circles showing landmark positions of the reference shape. To change the display format, select an option in the green **Deformation Display Format** box under the plot box, then click **Show Def. (Procrustes)** again. If your reference shape is not conveniently or conventionally oriented, you can call a tool to rotate it by clicking the **Reference Rotation Active** button near the bottom of the window. Now, when you click **Show Def. (Procrustes),** the plot box shows only the reference in small pink diamonds and a new window asks you to enter the value of the angle (in degrees) that the reference should be rotated. Positive values are counterclockwise rotations. Enter a small value like 10 or 15 and click **OK**. The reference will be rotated and you will be asked if you want to rotate it again. Simply click **OK** to rotate it the same amount again, or enter a new value. Reverse the sign if you want to go back. When you are satisfied with the rotation, enter **0** and click **OK**. Now, the plot box will show the deformation in the new orientation (it will be the same deformation). If you click **Show Def. (Procrustes)** again, you will see the reference in the original orientation, and the rotation entry window will show the net result of the rotations you performed last time. Click **OK** to accept this value, then **0** to obtain the same rotation. Click **Reference Rotation Active** a second time to turn this function off and use the original orientation.

To get a closer fit of the grid to the space occupied by the landmarks, you can change the value in the **Adjust Grid Size for PW** box, then redisplay the deformation. For an even tighter fit, click the **Grid Trimming Active** button. This will put two red boxes on the grid when you redisplay the deformation. These boxes indicate where the corners of the grid will be after trimming. First you will adjust the *x*-position of the lower left box, then its *y*-position, then the *x*- and *y*-positions of the upper right box. To do this, put the cursor on the screen to the left of the box and left click to move it in that direction. Put the cursor on the right side of the box and *left* click to move to the right. To fix the position of the box on this axis, right click. Now, to move the box up or down, left click above or below it, and again right click to fix it. Repeat for the other box. You cannot move the boxes in a different order, if you do not need to move a box in a particular direction, just right click to fix it where it is. After the fourth right click, the trimmed grid will be displayed, scaled up to fill the box. To turn off the grid trimming function, click the **Grid Trimming Active** button again.

There are several options for modifying the deformation drawings in the **Display Options** menu. In addition to changing line weight, symbol size and fill, it is also possible to choose red or black grid color, change the density of grid lines, select between a small set of options for the symbol to be drawn at the landmark positions, and choose whether or not to put arrowheads on the landmark displacement vectors. Note that all these options require you to redisplay the deformation before they take effect. That will also cause symbol size and fill to revert to their defaults, so plan ahead.

Figure W5.4A shows the deformed grids produced using default settings (except that the axes were removed) and saved as an uncompressed *.tif. In Figure W5.4B, the symbols were enlarged and filled and the line weight increased to 1.5. In Figure W5.4C, the number of lines was increased by setting plot density to 30 and the line color was changed to red. Note that in all images the deformed grid appears as stepped lines. To produce a vector graphic image from this display (Figure W5.4D), the figure was copied to the clipboard, pasted into a graphics program (where line weight was reduced after rescaling the image) and then saved as a *.tif.

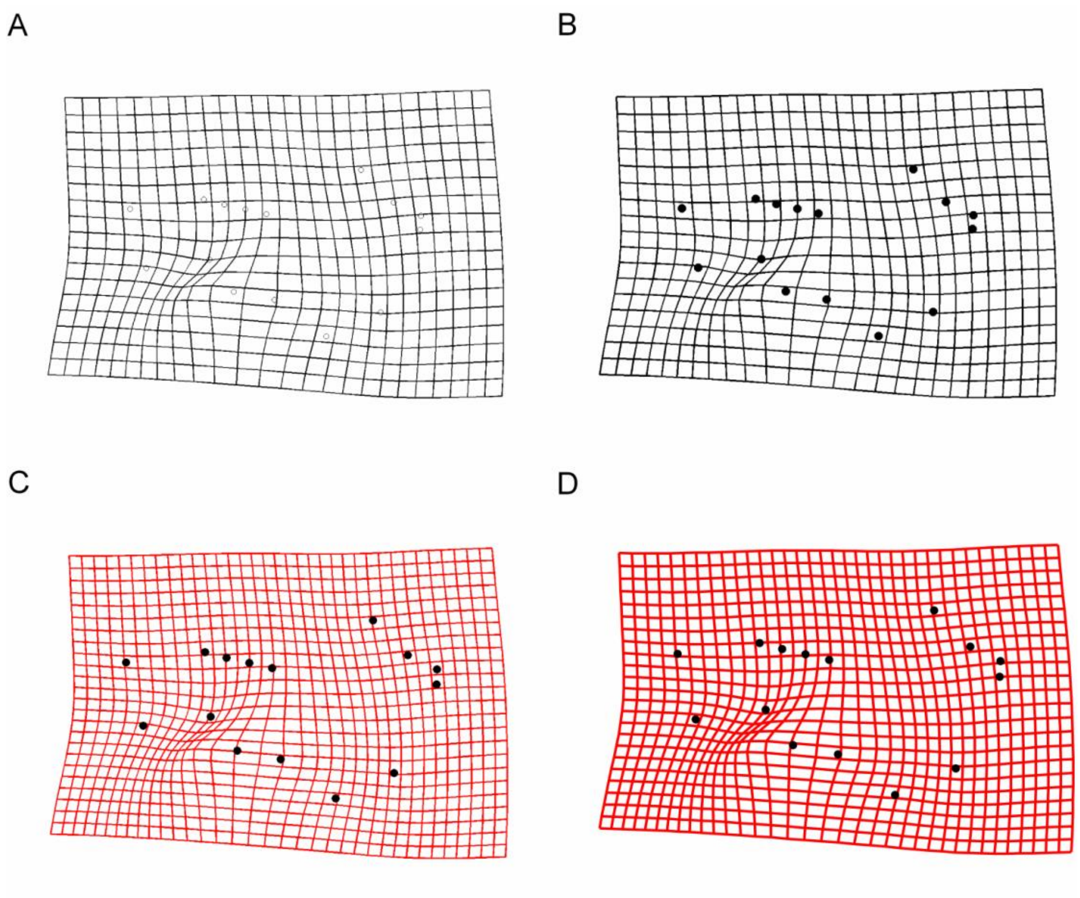


Figure W5.4. Deformed grids produced by **PCAGen** and saved as *.tif (A-C) or copied to clipboard and pasted into a graphics program (D). A: Default settings. B: Symbols enlarged and filled and lines thickened. C and D: Plot density increased and grid lines colored.

Simple3D7

This program serves the main functions of **CoordGen** (transforming between data-file formats) for 3D data. It also allows you to compare shapes of two different samples. Although the two **Load Data** buttons are labeled 1 and 2, you can load (and display) them in either order. You can also load the same data set into both. In most respects, graphic outputs for the 3D IMP programs are comparable to those of their 2D counterparts. However, 3D deformed grids are not implemented, which is due to difficulties in visual interpretation, not computation. Also, two features have been added to add visualization of shape differences: rotation and wireframes. To see what these features are, we will load a data set into **Simple3D7**.

Start **Simple3D7** and load *marmots3D.txt* as the data file. **Simple3D** requires you to open the 3D figure (left column of buttons) before displaying a data set. Other programs, like **ThreeDPCA7** will open the 3D figure automatically when you request a display. In **Simple3D**, you can open the window before or after loading a data set, when you do, a new window will pop up labeled "Figure #" (initially, it will be "Figure 1", but the number will increment if you keep opening new windows in the same session). The figure will remain blank until you click a Sh**ow Mean** or **Show Data** button.

After loading a data set and opening the 3D figure, click the **Show Mean** or **Show Data** button in the same column to see the shape(s) drawn with blue or red symbols. If you click one show button after the other, the two figures will be superimposed. To see only one data set or one mean, click the **Clear Figure** button before selecting the next thing to show.

Wireframes

The wireframe file is a text file similar to the "protocol" files you have already seen for other IMP programs. The first column numbers the wires from 1 to the end. The next two columns indicate which landmarks are to be joined by that wire – in *marmotwires.txt*, the first wire connects landmarks 1 and 2. The wires can be listed in any order, but you may find it easier to write the file if you think of connecting the points in an anatomically sensible order; perhaps one that is not far from the sequence of digitizing. For example, in *marmotwires.txt* the first three wires draw a triangle around the nasal opening; the next few connect that triangle to more posterior points near the eye. In general, the sequence proceeds from anterior to posterior along the dorsal surface, then back toward the anterior through ventral structures and teeth. You may also find it helpful to print several images of a specimen in different orientations, then draw lines on it to "highlight" the features that connect the landmarks. Because the wireframe does not enter into the analysis, you can always feel free to revise it if it isn't helping you to see the shapes and where they differ.

5. Visualizing Shape Change

To see the wireframe on one of the means, click the dark green **Load WireFrame** button, navigate to where you put the file and open it. Now click the **Show Mean** button. It should now be easier to see the marmot in the constellation of points, but you still may want to refer to a photograph. It may also help to click the **Label Landmarks** button. Landmarks 1-3 form the triangle around the nasal opening, 10-12 are the posterior edge, 16-18 are on the zygomatic arch.

The wireframe is only displayed with the mean, so to see it with the data, you have to click both the **Show Mean** and **Show Data** buttons for that data set. Similarly, you can see the difference between two means by clicking the show button for each data set. The symbols at the vertices of the wireframe show where the landmarks of the two means differ in location; however, the wireframes connecting the landmarks are always drawn as straight lines, so they are not good guides to interpolating the deformation between the landmarks.

The 3D figure toolbar

The toolbar for the 3D figure has the usual Windows buttons for New, Open, Save and Print. The **New Figure** button will open a new blank figure window with the same buttons in the tool bar. The **Open File** button will open the usual screen for finding a file to open, but notice that it is expecting a file with a *.fig extension. It will only open files with that extension. The **Save Figure** button will actually save the entire 3D configuration of points and wires if you use the default MatLab format (with the *.fig extension); when you open this file in a new MatLab figure screen, you will be able to manipulate it as you did the original. You may not be able to save it again, but then, you don't really need to. You can select other formats (and they will become the new default), but they will produce only 2D images of the perspective that is shown at the time you clicked the button. (You may have to open a new figure if you want to save more than one image.) The **Print Figure** button will open the usual printer control window for your system; you can print the 2D perspective you are currently viewing, without the gray background.

The second set of buttons on the toolbar of the 3D figure window allows you to change your perspective on the 3D data. The first two are magnifiers for increasing or decreasing zoom. Click on the button, then click on the figure to increase or decrease magnification incrementally. (For continuous zoom, just grab the edges of the window.) The third button (hand) allows you to drag the figure around in the frame, which is useful for bringing features back into view if they've been zoomed out of the frame. The last button in the second set is the one that allows you to rotate the figure. Click the button, then click and drag the image. In some programs, you will see a set of blue lines bounding the space to help you with orientation.

The next button is called the **Data Cursor**. It will change your cursor to a plus sign (+), and when you click on a data point in the figure, it will mark that point and bring up a box with the coordinates of that point. If you right click on the box you will open a menu for controlling this function. Changing **Selection Style** to **Mouse position** will allow you to get coordinates for points on the wireframe. Changing **Data Style** to **Window Inside Figure** will move the listing of coordinates out of the volume of the coordinates to a corner of the window frame. The default style is called a datatip; to compare coordinates for different locations click **Create New Datatip**, then click on the next point. And, of course, you can delete all of the datatips when you are done. The remaining functions are for manipulating the data in MatLab, so we will not discuss them here. The last button on the tool bar is intended to provide a legend for the figure, however, the list extends out of the screen for all but the smallest data sets, and it includes the wires as well as the specimens.

Other figure controls

Simple3D has a small set of simple figure manipulations available on the main screen. Removing the axis is probably not advisable, because the axes are your only cues to the 3D perspective of the figure. **Copy to clipboard** copies the control interface, which has lots of pretty colors, but no data (unless you have analytic results written at the bottom). The **Display Controls** menu on the toolbar has the familiar options for changing the size and fill of the symbols. The options to change line weight apply only to the wireframe.

ThreeDPCA7

The interface for **ThreeDPCA7** is very similar to that for **PCAGen**; the most notable difference is the addition of the **Load Wireframe** button in the upper right corner. In this program, as soon as you load your data, the 3D figure window will pop up and show the mean shape as locations of empty green stars. Slide this window out of the way and on the first screen click **Load Wireframe**. After loading the wireframe, click **No Group list**. The 3D figure will show clusters of black circles with the vertices of the wireframe at their centers (the mean shape). If you have loaded the marmot data sets we used before, you have probably noticed that now the skull is standing on end; the display is rotated 90° from the orientation in **Simple3D**. In **ThreeDPCA7**, as in **PCAGen**, you can load a grouplist (*marmotgrps.txt*) to label subsets in the data. (You don’t need to clear the figure, just select **Show Data** after you load the list.)

The controls on the 3D figure are the same as in **Simple3D**. If you used a group list, the legend will tell you which symbol goes with which individual in the data file. There are additional controls in the **Display Options** menu on the main screen, including options to change symbol size or fill. The option to increase line weight applies only to the wireframe, not the symbols. There also are a few controls in the **3D Axis Controls** menu: axis on/off and labeling for both the axes and the landmarks.

If you generate a scatterplot (**Show PCA Plot**) or a scree plot (**Statistics** menu), your only available manipulations are in the **2D Axis Controls** menu. This menu only provides the axis on/off and zoom functions. The **Display Options** menu does not apply to these plots.

The **Deformation Display Format** list, which determines what will be shown when you click **Display PCA Deformation**, has only three options. **Wireframe on Both** draws black and red wireframes representing the starting and target shapes. (Again, these help show changes in landmark positions; they are not guides to interpolating the deformation between landmarks.) You can change the target wireframe from red solid lines to black dashed lines by going to the **Display Options** menu and selecting **Use Dashed Line…**. The options for changing the symbols in this menu do not apply to the deformation plots, but the line thickness options do apply to the wireframes. The display format **Vectors on Landmarks** shows only the starting form, with vectors as blue lines (without arrowheads) pointing to the landmark positions of the target. These lines are also affected when you change the wireframe line thickness. The last display format option, **Wireframe on Target** shows only that ending configuration.

For exporting figures, there are separate buttons on the main screen for copying 2D and 3D figures to the clipboard or to *.eps files. As in **Simple3D**, there are many more export options for the 3D figure in the **Save** menu of that window.

tps Programs

tpsRelw

Because you have already been introduced to **tpsRelw**, we show here the display options for that program. The options for showing results in **tpsRegr** are nearly identical to those in **tpsRelw**. In **TpsTree**, there are somewhat larger differences in how you select what deformation to show, but these are so specific to that program, we reserve them for Chapter 10, when phylogenetic analyses are discussed.

After you load your data, the buttons to compute the consensus, partial warps and relative warps are enabled in sequence, as are the buttons to display those outputs. When you click one of the display buttons a new screen will appear showing the selected output. All of the plot windows can be expanded by grabbing and dragging the window frames.

Superimposition

Click the **Consensus** button in the **Display** box to open a new window. The default for this window is to show only the mean shape (consensus), with small open circles at the location of each landmark. As with the IMP programs, symbols for landmarks or data points must be substantially enlarged (more than seems adequate on the screen) to make presentable graphs for talks or publications.

You can show all of the superimposed individuals by going to this screen's **Options** menu and selecting **Points**. You can also show the individuals as deviations from the mean by selecting **Vectors** (if you selected **Points** earlier, you may have to de-select it to see the vectors). If you have semilandmarks, you can also choose to show them before and after sliding.

To change other features of the plot, select **Options**… from the **Options** menu. This brings up a new window with several groups of controls outlined in boxes; you can also get this window by right clicking on the graph. None of these changes will take effect until you click **Apply**, or **OK** (which also closes the options window). The **Points** box has options for increasing the size of the points (**Radius**), changing their **Color** and **Fill**, and also for adding the numbers of the landmarks (**Label points**) and adjusting the **Size** of the numbers. If you selected vectors (Arrows) or have links (a 2D wireframe), you can change their color and Width. You can also change the arrows from solid to dashed or dotted, but if you have a large data set you may not see anything but rings around the landmark means. The **Margins** box has controls for adjusting the distance of the plot from the edges of the frame, and the **Background** box allows you to change its color. To eliminate all of the changes that have been applied, go back to the plot window, and in the **Options** menu click **Defaults**.

To export the plot, one option is to go to its **File** menu and select **Save plot**. Here, you have a choice between *.bmp and *.emf formats. The other option is to go to the **Edit** menu and copy to the clipboard, again choosing between the same to formats (Copy as …).

Scatterplots

Both the partial warps and relative warps display buttons call up scatterplots of scores on axes for the selected space. In each, you can choose which axes of that space to plot. Each has an **Options** menu in the tool bar with two items: **Option** opens the options selection window (or right click), **Defaults** returns all features to their original values. The options window contains the same choices for **Margins**, **Points** and **Background** as before. The options for line width, style and color are now applied to the axes of the graph (we will explain **Sequence** later).

Shape differences

In each ordination plot, one option for obtaining pictures of shape changes represented by the scores is to click on the **still camera** icon. This calls up a circle at the origin of the plot and opens a **…visualization** plot window with a deformed grid drawing (obviously, showing no shape change at this point). With the mouse, drag the circle to a new location; its coordinates will be shown in the new box above the plot, and the shape change will be shown in the visualization plot window. You can also type coordinates into the box, which is a way to see deformations that are literally "off the chart".

The **Options** menu for the visualization plot has a new item, **Vectors**, which toggles between deformed grids and vectors at landmarks; there is no option for both. The vectors will only have arrowheads if they exceed a fixed minimum length. In the relative warp visualization, after you choose vectors, you can replace the default Procrustes superimposition with a resistant fit if you want that.

In these plots, you again have options to change the **Points** and **Background**; the main difference is that the line options apply to the grids. The vectors are the same color as the points, and you cannot change their width. Figure W5.5A shows the grids drawn with the default settings and saved as *.bmp. In Figure W5.5B, the symbols were enlarged and filled, the grid lines were made solid and black. **Margins** changes will move the landmarks, but will not directly affect the grid. The default is for there to be 15 lines (14 cells) evenly spaced between the right and left ends of the shape. An equal number of lines extends from the bottom as far up as necessary to get them all in. If the result does not fit your array of landmarks, change the number of **Y** lines up or down to get a better fit. Leave the grid square. To change the overall density, change the number of **X** lines (choose an integer between 2 and 20) then adjust the **Y** lines to fit. In Figure W5.5C, the points were moved in from the edges slightly by increasing the margins

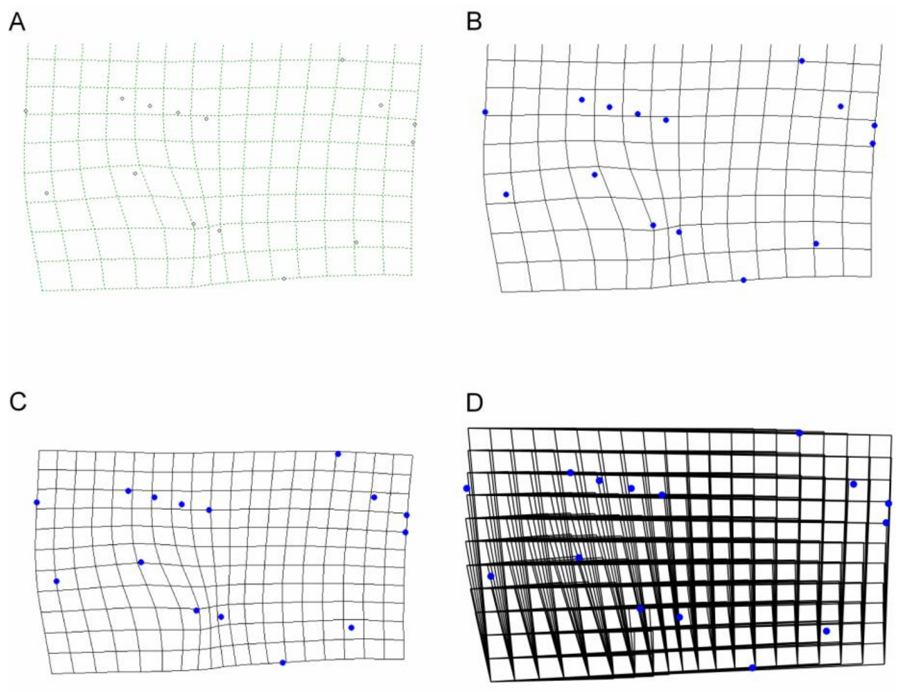


Figure W5.5. Deformed grids produced in **tpsRelw**, saved as *.bmp (A-C) or copied to clipboard as *.emf and pasted into a graphics program (D). A: Using default settings. B: Points enlarged and filled, lines solid and black. C: Margins and number of grid lines adjusted. D: emf coding of grid lines misinterpreted by a graphics program.

to 40; the number of **X** grid lines was increased to the maximum of 20 and the excess grid at the top was removed by reducing the number of Y grid lines to 12 – just enough to span the height of the jaw.

Images are the same whether you use the save buttons to create a file, or the copy to clipboard. In either case, the *.emf format used here is not always correctly read by graphics programs, as shown in Figure W5.5D.

Click the **video camera** to create a loop sequencing deformation at several locations in the plane. Now click on several locations in the space. New numbered circles will be added to the plot, with arrows connecting them in the sequence they were selected. The deformation for each location will be shown in the visualization window as they are added. There also are three new buttons on the toolbar for this window. The first will play the entire sequence one time. If you click the second button, the playback will keep looping through the entire sequence until you click the third button to stop it.

For the partial warps scores, the color used to draw the **Sequence** on the scatterplot can be changed using that plot's options window. After you click apply, the sequence will disappear. It will reappear in the new color if you click on the plot, but that will add another step to the sequence. To redraw the sequence, click the camera twice, to turn it off and back on. For the relative warps, the color cannot be changed, but the time between images in the sequence can be.

MorphoJ

Superimposition

After you have successfully loaded a data file into **MorphoJ** and performed the superimposition, it will open a graphics screen and show you the mean shape (large dots) and the variation around the mean (very small dots around the mean for each landmark). If you right click on this or any other graphics screen in **MorphoJ**, you will call up a list of options for that graph. In this case, the list of options is rather short: **Rotate** (90° increments), **Flip** (reflect) and **Resize** points (which only applies to the points for the mean). Export options for the graph are also listed in this menu, and are the same for any other **MorphoJ** graphic. In the window that opens, you must choose the **Type** first, then enter the name. If you enter the name before you choose the type, the name will revert to a default.

Scatterplots

Scatterplots of scores may be available for some analyses (click the tab labeled **Scores**); some bar charts will have no options. For scatterplots, you can choose which axes/variables to plot, if more than one is available, but you cannot change how the axes are drawn or labeled; for that, you will need to export it to another program. You can choose colors for the points as you did for landmarks in the shape comparison graphs. **Label data points** writes identifiers from the data file on the graph next to each point. Changing the size and color of the data points will not affect these labels. Other options made available when right clicking on a graphics screen are more specific to the type of graph and data. For example, if you chose a Principal Components analysis, you could set the scale factor for the deformation (the score that the deformation represents) or choose a different axis of deformation.

To distinguish between subgroups in the data, you will need to import a file with the classifier variable (equivalent to a group list in IMP except that the identifier for each individual must be the first column in the file). In most cases, you will have loaded this file in advance so you can do the analysis. After you load the list, you can choose colors for each group. When you have selected a color, as described above, you must click **Use Color** before going to the next group. Do not click OK until you are done selecting colors for all groups, because that will close the selection window. If you have a classifier variable, you will also be able to use it to draw ellipses around those groups in scatterplots. In the plot options list, select **Confidence Ellipses**. In the new window, you can choose between **Equal frequency ellipses** for the distribution of individuals in the group and **Confidence ellipses for means**. Below that, be sure to check **Use a classifier…** and also **Use this classifier to determine colors…**; that will insure the ellipses are the same color as the points for each group. Figure W5.6A (on page 19) shows a scatterplot of PC1 scores for the squirrel jaw data, saved as *.png; demonstrating that this image file format works well for these graphs.

Shape differences

For many analyses, one of the graphics tabs will have an image of shape transformation, it may even be the tab that automatically opens when the analysis is completed. Often (as in PCA and Regression), this window will show a deformation from the mean as vectors at landmarks – what **MorphoJ** calls a **lollipop graph**. Figure W5.6B shows a lollipop graph for squirrel jaws with the default settings saved as *.png; showing little or no difference from what is seen on the screen. The options menu for any deformation graph includes the option to change that type of graph; the other choices are **Transformation grid** (Figure W5.6C), **Warped Outline** and **Wireframe**. **Curves** and **wireframes** serve the same function of helping you to visualize the shapes.

For each of the four types of deformation graphs, there are plot options that can be set globally for that type of graph. Go to **Preliminaries**, then **Set Options for Shape Graphs**. Some options are shared among types of graph, but are still set separately for each type of graph (e.g. **Show landmark numbers** or **Show starting shape**). Most of these selections are rather obvious, but some features of color selection may be unexpected.

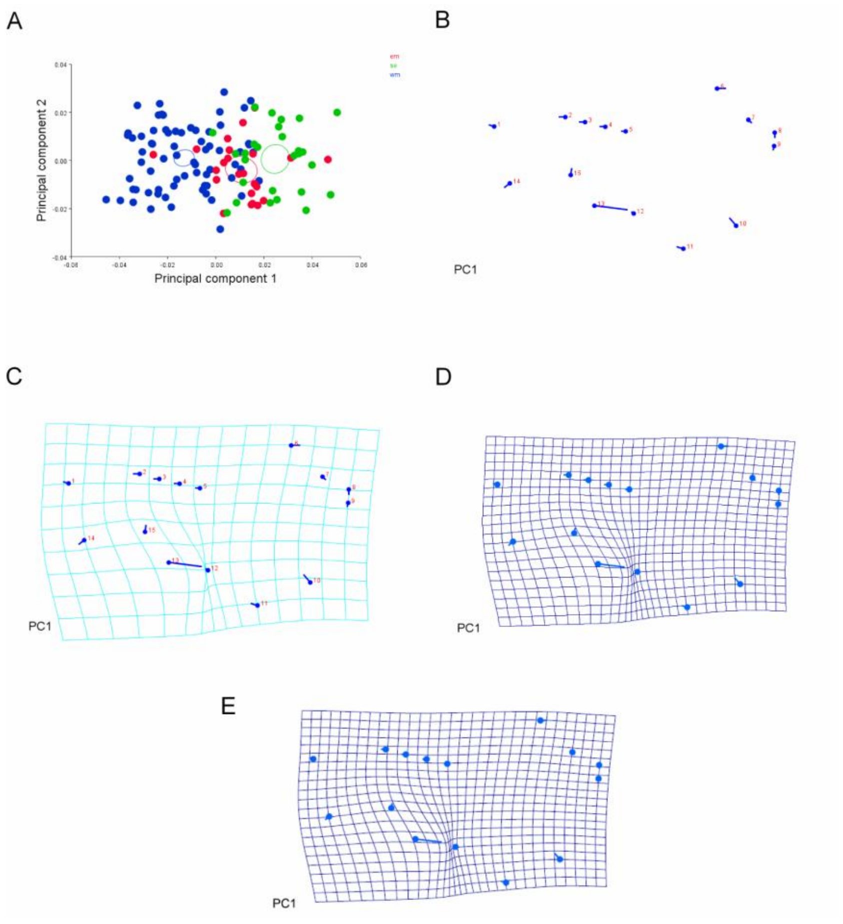


Figure W5.6. Output images generated using **MorphoJ**. A-D were saved as *.png, E was saved as *.pdf. A: Scatterplot showing PC scores color coded by geographic sample and ellipses of confidence intervals around sample means. B: Lollipop diagram showing displacements at landmarks, using default settings. C: Deformation grid with default colors and lollipop vectors added. D and E: Deformation with lollipops, darker grid color, reduced scale factor and doubled numbers of grid lines.

When you choose a color selection menu, a new window will be opened with a grid of "swatches". Choose one and it will be copied to the **Recent** grid and compared with the previous color in the **Preview** box. It is not used until you click **OK**. If you are not happy with the available swatches, you can go to the HSB or RGB tab and use the sliders there. In either case, the current color is shown in the **Preview** box. Click **OK** or **Cancel** when done. The choices you made are still not implemented. You must click **Accept** or **Cancel** at the bottom of the **Set options** window, which will also close that window. You still need to **redisplay** the graph to put the changes into effect.

For **transformation grids**, you can change **grid density** by right clicking on the graphics screen then selecting **Number of grid lines**. The default numbers will produce square grids for a score of 0. To be sure that your new grids are also square, you should check what the current values are for both horizontal and vertical lines, then change them proportionately. Figure W5.6D shows the deformation grid with scale factor reduced from 0.1 to 0.08 (to reduce the folding in the middle), doubled numbers of grid lines and darker grid lines. You can see in Figure W5.6D that the *.png format does not work as well for deformed grids as it did for the scatter plot (Figure W5.6A). To obtain smooth curves (Figure W5.6E), the image was saved as *.pdf and that image was copied into the graphics program from the pdf reader.

Outlines and wireframes

The **Warped Outline** option for deformation figures requires you to load a file with descriptors of curves to be drawn on the graph. This file can be created in **tpsDig** (e.g. with the draw curve tool) or in **ImageJ**. Whatever the source, it will require some editing to convert it to an acceptable format; we refer you to the **MorphoJ** manual if you are interested. The curve is treated as a background feature and is automatically deformed in accordance with the same thin-plate spline interpolation function that is used to draw the deformed grids.

The **Wireframe** option requires you to load a file created within **MorphoJ**. Both 2D and 3D wireframes can be used in this program. To create a wireframe, go to the **Preliminaries** menu and choose **Create or Edit Wireframe**. To create a wire, either drag a line from one point to the next (you won't actually see the line until you reach the second point), or choose the points to **Link** from the two lists on the left. (With a 3D data set, you will see two views of the landmarks; you can draw the wire in either, it will be reproduced in the other.) To delete a wire, select it from the list of links, then click the **Delete link** button. When done creating the wireframe, click **Accept**. The wireframe will be listed in the **Project Tree**, next to a little tetrahedron icon. By the default, the wireframe will be drawn as straight lines, however, if you choose a wireframe graph, the option to use a **Soft Wireframe** will be added to the list of graph options. In the soft wireframe, the wires are bent in accordance with thin-plate spline interpolation.

Graphics in R

R has remarkable graphical capabilities, which offer you the opportunity to control nearly all aspects of your figures, including the dimensions of your plot, the plotting symbols (and their sizes), their colors, legends, titles and even the layout of the figures. The three main types of figures discussed in this chapter - superimposed coordinates, scatterplots of PC scores, and varied depictions of shape differences - can all be made in **R**, and the output is publication-quality graphics. The process of making the figures is initially very time-consuming because, with the control that **R** gives you, you have a lot to absorb, a lot to decide, and a lot of experiments to do with the graphical parameters. But once you have scripts for th kinds of figures that you want, you can routinely reproduce them. However, it is unlikely that you will actually *want* to reproduce them because there are so many possibilities that you can explore. We cannot possibly survey them in this brief section. Instead, we touch on some of the possibilities and provide foundations for scripts that can serve as a starting point for depicting superimpositions, scatterplots and displays of shape differences.

In **R**, the graphics commands are distinguished as high versus low-level. High-level commands, such as **plot()**, create a new plot. You can have the entire plot made by that one command, which specifies the data to be plotted, the type of plot, the axes and the text and size of the labels for the axes, plus plotting symbol, symbol size and color and even the title. Low-level commands add to the plot. They may add lines, such as the vectors at landmarks, or the links between landmarks, or a palette of colors to fill the symbols, etc. Most of what you will work with are the low-level commands. There are some others, however, that you may find useful for setting up the dimensions of your plot, plus those that you need for saving the output to image files.

Superimposition

To plot the superimposed coordinates, you need to say what data should be plotted. When you do the superimposition in **R**, whether you use **procGPA()** from the shapes package (Dryden, 2012), or **pgpa()**, the function written by Claude (2008), both functions save the coordinates to ***$rotated**, where * is the name you assigned to the results of the superimposition. We will name ours **Procshape$rotated**. The *x*-coordinates are in the first column of the three-dimensional array, the *y*-coordinates are in the second, so you would plot **Procshape$rotated[,1,], Proc$rotated[,2,]**. Here we set up the plot using the **plot()** command, but the coordinates (and title) are added using the low-level plotting commands. To set up the coordinate system, and basic elements of the plot (like whether it has axes and whether the axes are labeled, and what those labels would be) you use the **plot()** command but specify that the type of plot is **type="n"**, meaning that nothing will be plotted. We also said that we do not want axes (**axes = F)** or labels, suppressing the default labels by putting no text between the quotes (**xlab=""** and **ylab=""**) and setting the aspect ratio of the plot to 1 (**asp=1**). The plot will be made using the low-level commands, **points()** and **title()**. The reason for making the plot using the low-level commands will become more apparent in the plots of the shape differences, when we want the other elements to be behind the landmarks. Because the low-level plotting commands add elements line by line, each written on top of what is already there, the first one is behind the second, which is behind the third, etc.

```
plot(Procshape$rotated[,1,],Procshape$rotated[,2,],type="n",asp=1,xlab="",
ylab="", axes=F)
points(Procshape$rotated[,1,],Procshape$rotated[,2,],pch=20,cex=2,col="black")
title("Ontogenetic change in a house mouse skull", cex=.75)
title(line=3,sub="ventral view",cex.sub=1)
```

The parameters that we specified are the plotting character, **pch=20** (a filled circle), the size of that character, **cex=2**, which means twice the default size, and **col="black"**, which means to make the symbol black. The title lines include the text of the title (both main and sub), the size of the text (0.75 times the default size for the main title, and the default size for the subtitle - that obviously did not need to be specified since it is the default value, as is the color "black", but having the parameters written out makes it easier to experiment with other sizes or colors. There are a lot of symbols that you can explore (Figure W5.7). The plot that we just made is shown in Figure W5.8.

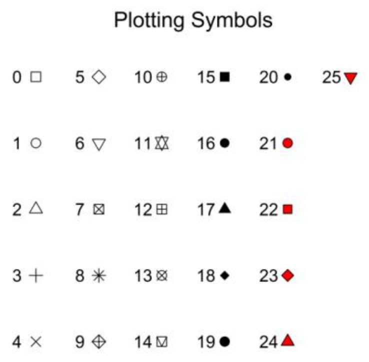


Figure W5.7. Plotting symbols in R.

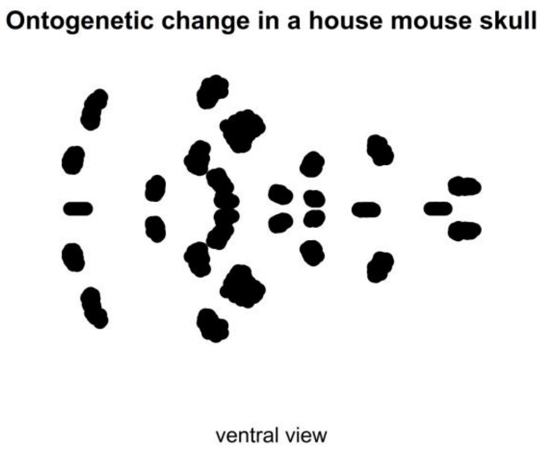


Figure W5.8. Superimposition done in R, output exported as a high-resolution *.tiff (1000 dpi).

You might find that the dimensions of the default plot window aren't suitable for your figure. You can change them, using **win.graph()**. You enter the desired dimensions, first the width then the height, and the desired font size. This creates a graphics window with those dimensions. You then ask for a new plot. For this plot, we used:

```
win.graph(5,4,12)
plot.new()
```

After you are satisfied with the dimensions of the plot and the plot itself, you can decide what image file type you want to save. Although it is possible to save the plot by going to the **File** menu and saving it as one of the image file types on the menu, that does not produce high-quality figures. Far better ones can be produced by writing a command for the device, inserting it right before the plot command. The output will be saved to that file rather than shown in the graphics window. When the output is saved, you shut off that graphics device. So the entire script for plotting the superimposition is:

```
win.graph(5,4,12)
plot.new()
tiff(filename = file.choose(), width = 5, height = 4, units = "in",
es = 1000)
plot(Dshape[,1,],Dshape[,2,],type="n",asp=1,xlab="",ylab="",
cex=2,axes=F)
points(Dshape[,1,],Dshape [,2,],pch=20,cex=2,col="black")
title("Ontogenetic change in a house mouse skull", cex=.75)
title(line=3,sub="ventral view",cex.sub=1)
dev.off()
```

We chose a high-resolution (1000 dpi) *.tiff file because it was required by Elsevier. There are several other possibilities, including *.jpeg, *.png*, .pdf and both *.cairo_pdf and *.svg based on cairo graphcs). To find out what is available, and to get help on graphical devices while in R, enter:

```
?grDevices
```

or you can do a websearch for "graphical Devices R".

Scatterplots

The plot that we just made is a scatterplot so the material covered in this section is not new. But, in this section, we will consider a scatterplot of scores on the first two components of a between-group PCA (see Chapter 6). In this case, we want to add new elements to the figure, such as color-coded symbols, varied symbols (and sizes) plus a legend (Figure W5.9). The variety of colors available in **R** is enormous. You can get a color chart at http://research.stowers-institute.org/efg/R/Color/Chart/. There are many shades of gray as well if you need gray-tones rather than colors (Figure W5.10). You can search for

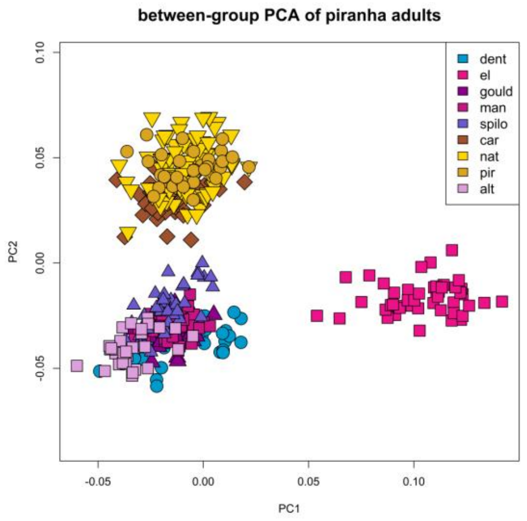


Figure W5.9. Scatterplot of scores for a between-group PCA in R, output exported as a high-resolution *.tiff (1000 dpi). The legend contains abbreviations for the species' names.

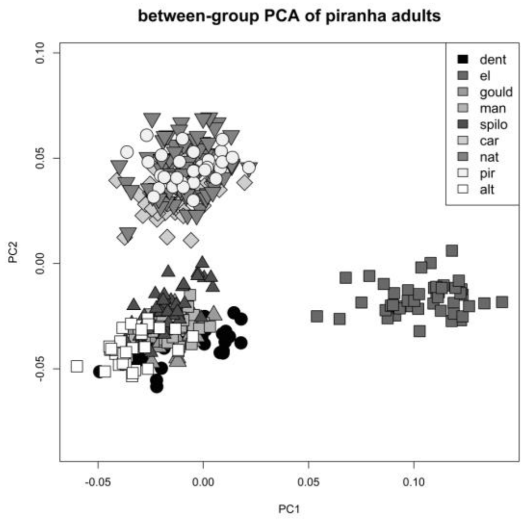


Figure W5.10. Scatterplot of scores for a between-group PCA in R; showing the variety of grays in R; output exported as a high-resolution *.tiff (1000 dpi). The legend contains abbreviations for the species' names.

colors if you can define a pattern that you want to match using the pattern-matching function **grep()**. For example, if you are looking for colors that are red (or reddish), you can enter

```
colors()[grep("red",colors())]
```

To make the plot, we wanted to vary both colors and symbols according to species. To be able do that without looking up which rows contain which species, we had a factor (here called "**species**") that identified each individual to its species. The data for this plot are principal components scores (actually, between-group principal component scores). We want to plot the scores on the first two components, which are contained in first two columns of the matrix E. To select the data for the first species, we need to specify the row indices for that species, but rather than hunting for those, we use **[which(species==1), 1:2]** which says that we want the first two columns of E for the rows of E for which species is "1".

```
plot(E[,1:2],type="n",asp=1,col="black",bg=(species))
points(E[which(species==1),1:2],pch=21,cex=2.5,col="black",bg="deep
skyblue3")
points(E[which(species==2),1:2],pch=22,cex=2.5,col="black",bg="deeppink2")
points(E[which(species==3),1:2],pch=24,cex=2.5,col="black",bg="dark magenta")
points(E[which(species==4),1:2],pch=22,cex=2.5,col=
"black",bg="mediumvioletred")
points(E[which(species==5),1:2],pch=24,cex=2,col="black",bg="slate blue")
points(E[which(species==6),1:2],pch=23,cex=2.5,col="black",bg="sienna")
points(E[which(species==7),1:2],pch=25,cex=2.5,col="black",bg="gold")
points(E[which(species==8),1:2],pch=21,cex=2.5,col="black",bg="goldenrod")
points(E[which(species==9),1:2],pch=22,cex=2.5,col="black",bg="plum")
legend('topright',legend=c('dent','el','gould','man','spilo','car','nat',
'pir','alt'),
fill=c('deep skyblue3','deeppink2','dark magenta','mediumvioletred',"slate
blue", 'sienna','gold','goldenrod','plum'),cex=1.2)
title("between-group PCA of piranha adults",cex.main=1.5)
```

We added one parameter to this script that we did not use in the superimposition plot (although we could have). The symbols that we used in this plot have two colors, one for the border of the symbol (**col**) the other for the fill (**bg**). The symbols numbered 21–25 can have different colors for the border and fill. In addition to the plot itself and the title, we added a legend using the **legend()** command, first saying where we wanted it to be (**topright**) and then what we wanted it to contain (abbreviations for the species names), and then the colors for the fills. This all takes a lot of typing, but it is easy to change one element at a time, such as the color for one species, its symbols or its symbol size.

Shape differences

We can show shape differences as vectors at landmarks (Figure W5.11A), vectors at landmarks plus lines connecting the landmarks (Figure W5.11B), a deformed grid (Figure W5.11C) and a deformed grid with vectors at landmarks (Figure W5.11D). We can also add a heat map to the grid, and add vectors at landmarks to that figure (Figure W5.12). We will go through these one by one, including the

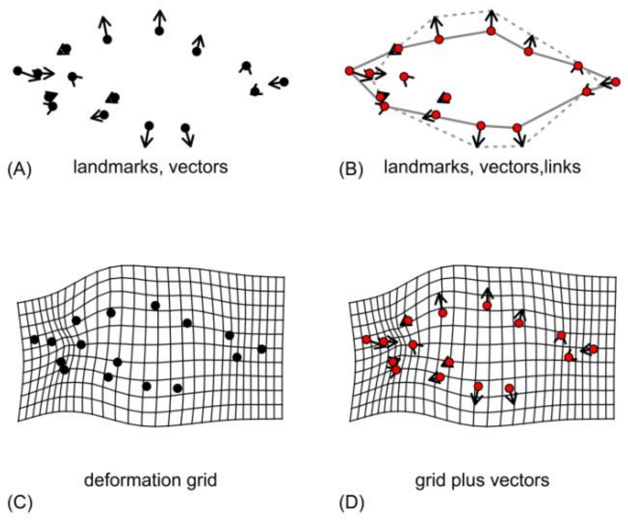


Figure W5.11. Depiction of shape differences in R. A: Vectors at landmarks. B: Vectors at landmarks plus links between landmarks. C: Deformed grid. D: Deformed grid plus vectors. The plot, including the layout of the four frames, the titles and the lettering of the four frames was all done in R. The output was exported as a high-resolution *.tiff (1000 dpi).

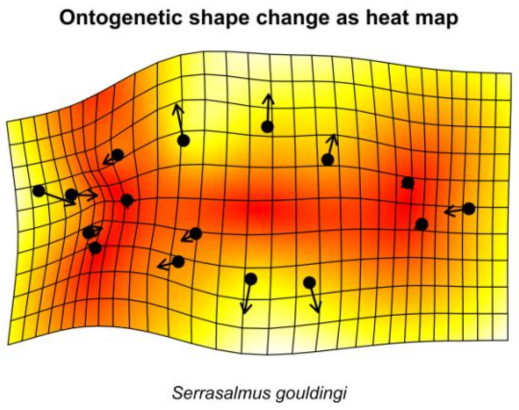


Figure W5.12. Depiction of shape differences as a heat map. using showDef() in R.

commands for laying out a figure with these four panels but, because the script is quite long, we just summarize the novel elements; the whole script is available as **FourPlots.R**.

To make these plots, you will need two shapes, the reference form and the target form. These are what you get from the statistical analyses. For example, if you do a regression of shape on size, you would show the shape change from the smallest (the reference) to the largest (the target). If you do a comparison between two groups, one would be the reference, the other the target. These plots are far easier to do if the *x*-coordinates are in the first column and the *y*- coordinate in the second. This is not the **shapes** format because it is a two-dimensional matrix rather than a three-dimensional array, but the data for an individual are organized like they are in the **shapes** format.

The first step is to plot the two shapes to see if you need to amplify the shape difference and/or rotate the data (to reorient the picture). Again we set up the plot using the **plot()** command and use the low-level commands, **arrows()**, **points()** and **title()** to make the plot. Here, the two shapes that we will plot are the reference shape matrix (matr) and the target shape matrix (matt).

```
plot(matr, asp=1, xlab="", ylab="",type="n", axes=F)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
points(matr, pch=21,col="black", bg="red", cex=1)
title(main="Ontogenetic shape change",line.sub=-1, sub="Serrasalmus
gouldingi",font.sub=3)
```

What this does is to draw arrows from the *x*,*y*-coordinates of the landmarks in the reference form, which are in column 1 and column 2 of **matr** (**matr[,1], matr[,2]**), to the coordinates of the target form (**matt[,1], matt[,2]**). It then adds the landmarks of the reference form, as circles (**pch=21**) bordered in black (**col=black**) and filled in red (**bg=red**). The symbol size is the default size (**cex = 1**).

If the picture needs to be rotated, you can rotate the shapes (both of them) using **rotateAMatrix()**, by Adam Rountrey. The arguments for this function are the matrix to rotate, the angle of rotation and the coordinates for the center of rotation, so to rotate **matr** 180 degrees around the centroid, you enter:

```
rotateAMatrix(matr, 180,0,0).
```

You would do the same for **matt**. That rotation was needed for this figure. If you need to amplify the amount of shape change, you multiply the difference between the two forms by whatever amount you want to magnify the change by and add that to the target form. Here we create the variable **vecMag** (which is our multiplier). We then calculate the difference between the two shapes (**matt-matr**) and add that to the coordinates of **matt**.

```
vecMag = 3
matt1<-matt+(matt-matr)*vecMag
```

No such magnification was needed for this case.

The next plot (see Figure W5.11B) connects the landmarks of the reference form with a solid line in a darker gray and those in the target form with a dotted, lighter gray line. The code comes from Claude's *Morphometrics in R*. We connect the landmarks from 1 to 10 (with 1 connected to 2, 2 to 3, 3 to 4, etc). This is assigned to **joinline**. We then use the **lines()** command to draw the lines, putting **joinline** as the row index for each matrix. The lines, like symbols, can have colors. **lwd** is line weight, and **lty** is line type; **lty=3** is a dotted line connecting those landmarks. We add the points last so that the lines and vectors are behind the landmarks.

```
plot(matr, asp=1, xlab="", ylab="",type="n", axes=F)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
joinline<-c(1:10,12,1)
lines(matr[joinline,],col="grey50",lwd=2)
lines(matt[joinline,],col="gray59",lty=3,lwd=2)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2.5)
points(matr, asp=1,pch=21,col="black",bg="red",cex=1.5)
```

The last two plots (see Figure W5.11 C and D) use deformed grids to show the shape changes. There are several functions that will do the grids: **tpsgrid()** in the shapes package by Dryden, **tps()** by Claude, and **myTps()** by Adam Rountrey. The three functions differ in their arguments and outputs. Because they all will work on the same data, you can experiment with all three. To make these figures we used **myTps()**, which allows us to control the degree to which the grid extends beyond the data (as well as to control the aspect ratio of the grid). This function depends on Claude's **tps2d()** so that will have to be read into **R** if you did not already source his code.

The first two arguments are the two matrices,

```
myTps<-function(matr, matt, n, expandX, expandY, show=TRUE)
```

the reference and target matrices (**matr** and **matt**) respectively. The next is the grid density (**n**), i.e. the number of horizontal/vertical lines in the grid. The next two, **expandX** and **expandY**, control how much the grid extends beyond the range of the data. **expandX = 1** means that the grid will go out 1 times the *x*-range of the data to the left and right and **expandY** means the same for the *y*-range. Usually, you will want far smaller numbers than those. The last parameter is whether to show the grid (the default is to do so). You might not want to do that if you are running this function in order to use **showDef()**. The input for Figure W5.11C is:

```
grid=myTps(matr,matt,30,.3,.0,show=TRUE)
```

We can add the landmarks and vectors to the plot to produce Figure W5.11D:

```
grid=myTps(matr,matt,30,.3,.0,show=TRUE)
#draw the arrows
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2.5)
#draw the points
points(matr, asp=1,pch=21,col="black",bg="red",cex=1.5)
```

To make the final figure (Figure W5.12), we use Adam Rountrey's function **showDef()**. This takes the output of **myTps()** (here saved in the variable **grid**). Because it uses **myTps()**, it also depends on Claude's **tps2D()**, and it also requires the package **sp** (Bivand et al., 2008; Pebesma et al., 2005). The function will load it, but you need to have installed it.

```
showDef(grid)
points(matr, pch=20,cex=2)
arrows(matr[,1],matr[,2],matt1[,1],matt1[,2], length=0.07,lwd=2)
title(line=-1,"Sexual dimorphism in gorillas",cex.main=1.5)
title(line=-1,sub="From female to male; magnified by 3",cex.sub=1.2)
```

There are other color mappings. To see what is available, and for more detailed information on the graphical parameters in R, you can get help in R by entering:

```
?graphics
```

There is also a wealth of information online about graphics in R. Do a websearch for "graphics in R".

Laying out multipaneled plots

To combine the four panels of Figure W5.11 in a single plot, we first set the margins (using **mar()**), to be one line on all four sides. We then positioned the first figure to extend from 0.0 to 0.5 of the *x*-axis and 0.7 to 1 on the *y*-axis. These axes are scaled from 0 to 1 from the bottom left corner, so the frame occupies the left half of the width of the plot, and the top 3/10ths of the height; the deformation grids on the bottom of the figure take up more vertical space so they were allotted more than half the height of the plot. The **new=TRUE** means that each is a new plot; they are not additions to an existing plot. Because commands such as **arrows** and **points** are low-level plotting commands, these would usually be added to a plot. The titles are positioned above and below each frame, but to label the panels A, B, C, and D we added marginal text **(mtext()**.

```
par(mar=c(1,1,1,1))
par(fig=c(0.0,0.5,.7,1), new=TRUE)
plot(matr,asp=1,xlab="",ylab="",pch=21,type="n" ,axes=F)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
points(matr, pch=1,bg="black",cex=1)
title(line=-0.1,sub="landmarks, vectors")
mtext(adj=0,"(A)",side=1)
```

```
par(fig=c(0.5,1,0.7,1),new=TRUE)
plot(matr,asp=1,xlab="",ylab="",pch=21,type="n" ,axes=F)
joinline=c(1:10,12,1)
lines(matr[joinline,],col="gray50",lwd=2)
lines(matt[joinline,],col="gray59",lty=3,lwd=2)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
points(matr,pch=21,col="black",bg="red",cex=1)
title(line=-0.1,sub="landmarks, vectors,links")
mtext(adj=0,"(B)",side=1)

par(fig=c(0,.5,.2,.7), new=TRUE)
grid=myTps(matr,matt,30,.3,0)
points(matr,pch=21,col="black",bg="black",cex=1)
title(line=-1,sub="deformation grid")
mtext(adj=0,"(C)",side=1)

par(fig=c(.5,1,.2,.7), new=TRUE)
grid=myTps(matr,matt,30,.3,0)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
points(matr,pch=21,col="black",bg="red",cex=1)
title(line=-1,sub="grid plus vectors")
mtext(adj=0,"(D)",side=1)
dev.off()
```

Literature Cited

Bivand, R. S., Pebesma, E. J. & Gomez-Rubio, V. (2008) *Applied spatial data analysis with R* . Springer, New York.

Claude, J. (2008) *Morphometrics with R*.. Springer, New York.

Dryden, I. (2012) shapes: Statistical shape analysis. R package. pp.

Pebesma, E., Bivand, R., Rowlinson, B. & G'omez-Rubio, V. 2005. Classes and methods for spatial data in R. *R News 5 (2), http://cran.r-project.org/doc/Rnews/.*

6

Ordination Programs

Three methods are described in the ordination methods chapter (Chapter 6) of the textbook: principal components analysis (PCA); canonical variates analysis (CVA); and between groups PCA (bgPCA). The IMP package has four different programs for analyzing ordinations, two for PCA and two for CVA, according to whether the data are 2- or 3D. **PCAGen7a** is for PCA and bgPCA of 2D data, **ThreeDPCA7** is for PCA (but not bgPCA) of 3D data. **CVAGen7b** is for CVA of 2D data, and **ThreeDCVA7** is for CVA of 3D data. The interfaces all have the same basic layout. **MorphoJ** can perform PCA and CVA on both 2D and 3D data; PCA is found in the **Variation** menu and CVA is found is in the **Comparison** menu. **tpsRelw** (v. 1.49) can perform PCA on 2D data; that is the default form of relative warps analysis. **R** can do PCA, bgPCA and CVA (on both 2- and 3D data). Because the interfaces and the procedures used to run PCA and CVA and bgPCA, when available, tend to be similar within packages, this chapter is organized by package rather than methods.

Data input, the most common features of the graphic displays, and the process of exporting results have all been described in earlier chapters. Here we focus on the steps needed for requesting the analysis. We begin with PCA of 2D data in the IMP package, describing in some detail the steps needed to obtain the analysis. We then describe what you need to do differently to obtain a bgPCA for 2D data. This is followed with descriptions of 3D PCA, 2D CVA and 3D CVA, focusing on their differences from 2D PCA. The descriptions of ordinations in **tpsRelw** and **MorphoJ**, which follow in that order, are relatively simpler. In **tpsRelw**, there are fewer kinds of ordinations available because **tpsRelw** does only PCA (with options for relative warps analyses that are not PCA). In **MorphoJ**, you indicate when loading the data whether they are 2D or 3D so the only distinction between the analyses lies in whether you do PCA or CVA. In **R**, conducting PCA and bgPCA are very straightforward but the process of producing the two shapes whose difference is depicted graphically is less so. Thus, in this chapter, we focus on the steps taken to obtain the analyses in the specialized morphometrics software and on the steps needed to conduct the analyses and to produce the reference and target shapes in **R**.

IMP Programs

PCAGen7a(2D)

Entering Data

Start the program and load the data (*weslm.txt*) as described in earlier chapters. If you want to have groups labeled in the plots of either the superimposed specimens or PC scores, click **Load Group Membership List** (*wes groups.txt*). Find and select the file with the group codes. If you don't have a group list, or don't want to use one, click the **No Group List** button. (For directions on making a group list, see Chapter 5.)

After the group list has been loaded or you selected not to load one, a Procrustes superimposition will be performed and the superimposed shapes will be written to the display box. (For *weslm.txt*, there are 15 clusters around the average position of each landmark.) It does not matter what superimposition you did prior to loading the shape data because a Procrustes superimposition is automatically performed when you load your data into **PCAGen**. You can use the buttons for other superimpositions to show what they look like but they will have no effect on the PCA because that is always based on the Procrustes superimposition.

Getting numerical results

Click the **Show PCA Plot** button in the purple box (yes, the analysis has already been done; in fact, it was done before the superimposed shapes were plotted). The plot now shows the positions of all specimens in the space of the first two PCs. The proportion of the total variance each PC represents is listed in the boxes under **Variance Explained**. To see where individual specimens are located, click the **Label Points on PCA Plot** button and click again. To see the locations of sample means, click **Show Means**, then **Show PCA Plot** and look for the large symbols (if you did not load a group list, there will be only one, at [0, 0]). To see other axes, click the up and down buttons, and then click **Show PCA Plot** again.

The **Axis 1 +** and **Axis 2 +** buttons reverse the signs of the scores plotted on the *x*- and *y*-axes, respectively. This is easiest to see if you first turn on the **Label Points …** button, display the plot, then reverse one of the axes and display the plot again. If you are using group codes, it will be pretty obvious when groups switch sides of the plot, otherwise look for the numbers labeling individuals at the edges of the distribution. The signs of principal components are entirely arbitrary, so different programs, and even different iterations with the same program can produce plots with reversed polarity (everything else will be the same). Consequently, these toggles can be useful for enhancing the comparability of plots between programs; they may also be useful for comparing results for different data sets.

Exporting numerical results

To save the PC scores, go to the **File** pull-down menu and select the first item, **Save … Matrix**. Use the pop-up window to navigate to the folder where you want the text file saved and to give it a name. You can also do this by clicking the **Save PCA Scores** button in the orange box. Also in the **File** menu is an option to save the eigenvalues of the PCs. These are ordered from smallest to largest.

Determining the number of distinct components

Use the **Statistics** pull-down menu to generate scree plots of the eigenvalues or percent of variance. The eigenvalues are the variances, so these plots differ only in scale. If you saved the eigenvalues, you have a list of the same values in reverse order. This list spans the entire 2K-4 dimensions, but the eigenvalues probably become indistinguishable from zero well before the end. If you open the list in a spreadsheet, you can compute the % variance and cumulative % variance to determine how many PCs you need to account for a given proportion of the total. You can also determine how many axes account for a fraction of variance larger than 1/(2K-4), i.e. the amount that 1 variable would account for if they all accounted for the same proportion. To run Anderson's test to evaluate whether consecutive pairs of eigenvalues represent significantly different amounts of variance, go to the **Statistics** menu and select **Significant Differences in PC Components**. The test progresses through successive pairs of eigenvalues until reaching a pair that are not significantly different. The pop-up window will then report the number of distinct eigenvalues (see text for details).

Note that the test will stop at the first pair of eigenvalues that are not different, so you will not know if the next pair of values are different or not. The PCs 1 and 2 might represent 50% and 40% of the data and so are not significantly different. But even if these axes are not distinct from each other (or from any other direction in that plane), they may both be distinct from the PC3. Thus, it may be beneficial to look beyond the numerical result of the test to the shape of the scree plot for a more complete understanding of the distribution of individuals in the sample.

Interpreting PC scores as shape differences

Now that we have scores, we want to see the shape changes they represent. To see the shape change from the mean represented by a positive score on the first axis, click **Show Def. (Procrustes)** in the silver box. The default scaling factor is 0.1, so you will see the deformation associated with a score of

0.1 on that axis. Changing the number in the **Scaling Factor** window will magnify or diminish the deformation shown when you redisplay the deformation. Increasing the scale factor is useful with the actual shape differences are too small to be seen easily; however, too large a value may also produce a picture that is difficult to interpret. Decreasing the scaling factor from the default value may be useful when the default is outside the range of observed values. You can also enter a negative number to see change from the mean in the opposite direction.

Combining deformations from two PCs

Up to now, we have shown shape differences from the reference along just one of the axes. Sometimes it is useful to show shape changes in other directions or differences between shapes that are not on the axes. These images can be generated with the buttons in the brown **Show Deformation implied by PCA** box in the lower right corner of the screen. First, call the plot of PCA scores. Now, specify that you want to see the results in Procrustes superimposition by selecting that option in the large white window in the brown box (the default is Bookstein shape coordinates). Clicking the **Place M1** button will cause a pair of cross-hairs to appear on the **PCAGen** screen. Center these over (0, 0) in the PCA plot. Now left click; this will put a red dot at the origin. Now click **Place M2**, put the cross-hairs on a point on the *x*-axis (the horizontal cross-hair should pass through the red dot), and left click to place a green dot on the axis. Clicking **Show M1** will show the deformation for a shape at M1, which would be no deformation in this case. Clicking **Show M2** will show the same pattern of deformation along the *x*-axis that you produced before. If the deformation seems small, increase the number under **Marker Exaggeration**. Clicking **Show M2-M1** will show the difference between the deformations. In this case, it is the same as the deformation for M2 because M1 was placed at the mean (so there is no deviation from the mean along this axis).

If you call the PCA plot back and click **Restore Markers**, the red and green dots will appear in the same places as before. Click the **Place …** buttons to put a second set of dots on the PCA plot displaced the same distance up the Y-axis from the originals. Now **Show M1** will reveal the shape change along PC2. **Show M2** will reveal a deformation that combines changes along PC1 and PC2 in proportion to the scores at the new M2 marker. And **Show M2-M1** will reveal the deformation along PC1 that you saw before, but mapped onto a different starting form – the position of M1. Note that the combination of PC1 and PC2 amplifies some features of the PC1 deformation and reduces others.

These manipulations can be used to show deformations in a plane defined by any two PCs; simply select the PCs to be shown in the PCA plot before placing the markers. However, it is important to keep in mind that these plots will only show deformations in the selected plane.

Between Groups PCA in PCAGen7a

To run this analysis, first find the button at the bottom of the screen, **PCA of Means** (below the **reference rotation active** and the **grid trimming active** buttons). Click that *before* you load the group list. If you have already loaded a group list, click **Clear Data + Reset**, then **PCA of Means** and then reload the data. You will get the same plot of superimposed specimens as with the ordinary PCA, but the variances explained will be different because this analysis is now being performed on the means. To see the mean shapes, go to the **More Plots** menu and select **Show Means** for the superimposition of your choice. **Show PCA Plot** will show the scores of the individuals on the axes that describe the differences between the means. To show the differences between means as deformations, use the deformation display options as before; these will now be calculated for the bgPCs, the axes describing the variation of the means. If you want to calculate the score for each mean on one or more of the axes, you can compute it from the scores for all the individuals. Click **Save PCA Scores** to generate a file with all scores for all specimens. Open this in your spreadsheet and compute the mean for each group on the axis of interest.

ThreeDPCA7 (3D)

ThreeDPCA, unlike **PCAGen7,** allows you to use a wireframe to visualize shape variation. To see what a wireframe looks like if you are interested in 3D analysis but don't yet have your own 3D data, use the files we provide: (1) *marmots3D.txt*, a file of 3D coordinates; (2) *marmotwire.txt*, a wireframe, and *marmotgrps.txt*, a group list. Load the data, wireframe and group list (if you want to see the groups). The shapes will be superimposed and the PCA done and the superimposed coordinates will be shown with a wireframe connecting the landmarks in the mean shape. The wireframe will also be used to illustrate shape changes. The wireframe can be shown on the reference (mean) with red dots illustrating the positions of the landmarks in the target by going to **Deformation Display Format** and selecting **Wireframe on Reference**. Selecting Wireframe on both will add a second wireframe in red connecting the red dots. If you select **Vectors on Landmarks**, the reference and its wireframe will be drawn, with vectors from those landmarks to the positions of the corresponding points in the target (which are not explicitly shown). **Wireframe on Target** will show only the target landmarks connected by the wireframe; the reference will not be shown. Note that every time you click **Display PCA Deformation**, the 3D plot reverts to its default orientation.

There are fewer options for manipulating graphs in **ThreeDPCA** than in **PCAGen**, and the axis controls have been placed in two menus for the 2D and 3D figures. The 2D figures include scree plots and the scatterplots of scores on pairs of axes, e.g. PC1 vs PC2. The 3D figures include the depictions of 3D superimposed data and the deformations. Another distinction between the programs is **ThreeDPCA** does not have an option to save or export the eigenvectors or eigenvalues (there is no **File** menu corresponding to the one on **PCAGen**). To record the % variance explained by any factor, scroll through the PCs using the **Up** and **Down** buttons in the **Show PCA Plot** box and jot down the number that appear under **Variance Explained**.

CVAGen7b(2D)

A new window

When you open **CVAGen7b**, the first thing you will notice is a new window, the **Auxillary Results Box**, which opens in front of the main interface. Minimize it or slide it out of the way and load your data file and group list. Unlike **PCAGen7a**, **CVAGen7b** needs the group list and will not run without it. Various tabular results will be shown in the Auxiliary Box, which can be copied to a text file by clicking the blue **Append**… button, or by using your mouse and keyboard to drag-select and then copy to the clipboard.

Datafiles with more coordinates than specimens

If you have more coordinates than specimens, which is likely if you have many semilandmarks, you will not be able to perform CVA on the coordinates without reducing their dimensionality. The easiest solution to this problem is to perform a PCA first, and then perform the CVA on the PCA scores. Although **CVAGen7b** cannot take PC scores as input, it can do a PCA before doing the CVA. However, you need to select this option before loading the data, and also to determine how many PC axes should be used as input for the CVA. Keep in mind that the CVA results may vary as a consequence of the number of PCs included; so you need to decide in advance how many PCs to use. Doing the PCA first in **PCAGen7a** can help you make that decision because you can examine the scree plot to see how many PCs it takes to explain the variation; if, for example, you want to use enough PCs to explain 95% of the variation, you can use **PCAGen7a** to figure out how many PCs that will take. Now start **CVAGen7b**, and *before* loading the data, look for the small box at the bottom left of the interface, which says **Use PCA Reduction**. Click that, and enter the number of PCs you want **CVAGen7b** to use. Then enter the data in the usual manner. If you already tried to load data but had forgotten to select the PCA reduction, just click the **Clear Data + Reset** button to start over. If the PCA reduction hangs up, try changing group codes. If the codes are not in numerical sequence, without any omitted numbers, the PCA reduction will not work. So if you left out a “2” from your group list because you do not like the symbol associated with that number, the PCA reduction procedure will read that to mean that the sample size for group 2 is zero.

Exploring results

When the computations are completed, the superimposed shapes will appear in the plot window as they did in **PCAGen7a**. A pop-up window will tell you how many significant CVs there are – this result is obtained by first performing a MANOVA using scores on all axes, then using all axes but the first, then also excluding the second, and so on until getting a non-significant result or running out of axes. Click **OK** to close this window. The *p*-values from the MANOVA test as well as Wilks' Lambda and the associated chi-squared value also are reported in the **Auxillary Results Box**.

Click **Show CVA Plot** to get a scatterplot of the scores. **Eigenvalues** of the CVs are displayed next to the axis selection buttons, where the proportion of variance was displayed in **PCAGen7a**. These values reflect the amount of between-group variation, scaled by within-group variation.

The silver deformation display buttons produce the pictures of shape change along each axis, as they did in **PCAGen**, with an important difference. If the **Regr?** (regression) button *is* pushed, as it should be when you download the program, the deformation shows the changes in the coordinate space that are *correlated with* the CV scores. These are the differences among the means, not the actual canonical variates. If the **Regr?** button is *not* pushed, what you will see is something very different **-** the variables that maximally discriminate the groups, i.e. the canonical variates. The pictures will exactly match the deformations seen if placing the markers (brown buttons) on the equivalent scores. This difference reflects that fact that CVs are not orthogonal in the space of the coordinates.

Testing the significance of group differences

To determine whether the means differ, go to the **Statistics** menu and select **Single Factor Permutation Manova -** this is a test of the null hypothesis that the means do not differ. It generates a conventional ANOVA table with sums of squares and the *F*-ratio, which are reported in the **Auxillary Results Box**. To test whether this *F* is statistically significant, assignments are permuted and the *F* computed for each permutation. The frequency of permutations with *F*- values higher than that for the *a priori* assignments is reported as the *p*-value. (See Chapter 8 for further discussion.) To evaluate how effectively groups are discriminated, go to the **Statistics** menu and select **Show Groupings by CVA**. In this assignment test, each specimen will be classified as belonging to the group with the closest mean; the numbers of specimens that are correctly and incorrectly classified will be displayed in the **Auxillary Results Box**. Each row lists the number in that *a priori* group that are closest to their own group mean (column with the same group code) and the numbers that are closest to the mean of a different group (other columns). These are resubstitution rates of assignment and involve a degree of circularity as the same data were used to create the axes and assess their performance as discriminators. Additional details can be obtained by selecting **Run Assignment Test** or **Run Detailed Assignment** test. Each produces a text file that reports each specimen's *a priori* assignment based on the group list (column 3) and its *a posteriori* assignment based on distances from the group means (column 4). In the less detailed output, the distance to the nearest group mean and the probability of that assignment are reported for each individual. In the more detailed output, distances to each group mean, and the probability of the assignment to each group, are reported for every specimen. In this report, you can see that some specimens cannot be definitively assigned to any one group – "Unique Group=0". This does not mean it is equally close to both groups; only that it is close enough to each group to be classified in either.

To run a simple jackknife cross-validation test, select **Jack-knife Groupings**. In this test, one specimen is left out of the data set, the group means are recalculated, and the specimen classified according to its distances from the recalculated means. After this is repeated for each specimen, the results are shown in the **Auxillary Results Box**. A more elaborate cross-validation test can be performed by selecting **Jack-Knife Assignment Test**. In this analysis, a fraction of the specimens are set aside as a test set, and the remainder are used to construct axes, which are then used to classify the test set. The pop-up window asks you what fraction to leave out as the test set (0.1 = 10%), and how many times to repeat the analysis. The results, reported in the **Auxillary Results Box**, list the total number of "unknowns" (number left out multiplied by number of repetitions) and the rates of correct and incorrect classifications for those unknowns.

ThreeDCVA7 (3D)

Like **CVAGen7b, ThreeDCVA7** has an option to perform a PCA reduction before doing the CVA and, as in **CVAGen7b**, you must select that option before loading the data. Also, you must load a group list. Loading data into **ThreeDCVA7** is otherwise identical to loading it into **PCAGen7a**. As in the other three IMP programs, the analysis is automatically performed as soon as the data are loaded.

When the analyses are complete, the number of distinct axes is reported on the pop-up window and in the **Auxillary Results Box**. Write these results down or copy them to a file before you close this window; you will not be able to retrieve these results after you close the window and box. The only option currently available from the **Statistics** menu is to show the distance based groupings, without cross-validation tests (this analysis also is not available after the **Auxillary Results Box** is closed).

This program also has fewer options than the others for graphical output. Only the Procrustes superimposition is available and there is no option to perform the regression. The graphic produced by clicking **Display CVA Deformation** is the same as you by get placing marker M1 on the same axis at the score corresponding to the **Scaling Factor** (or by adjusting the **Marker Exageration** so the score and the scaling factor match).

tps

tpsRelw

Open **tpsRelw** and load data by clicking the **Data** button and locating your data file. Go to the **Options** menu and make sure that **Alpha=0** is checked; that is the default and it is the value that produces a PCA. Other values differentially weight the partial warps. Also make sure that the option to include the uniform component is checked; that too is the default. Click each of the three buttons in the **Compute** window as they become active. The **Consensus** button will superimpose the specimens and compute the average shape (the consensus). The **Partial warps** button will compute the partial warps of the consensus (axes of the tangent space) and the scores of the specimens on those axes. The **Relative warps** buttons does the PCA. When that is complete, go to the **Display** box and click **Consensus** to show the reference shape in a new window. In the **Options** menu of this new window, click **Points** to show all the specimens in Procrustes superimposition. Go back to the first screen and click **Relative warps** in the **Display** box. A new screen will appear showing scores on PC1 and PC2. Clicking the camera icon places a circle at the origin of the plot and opens yet another window that shows the deformation. Drag the circle along the axes to show the deformations in each direction; drag it off the axis to show the combination of deformations along two PCs. As in **PCAGen7a**, these deformations only show shape change in the plane of the 2 PCs. Use the **X** and **Y** indices at the top of the plot window to select a different pair of PCs to plot and visualize.

From any window, you can go to the **File** menu and select **View report**. Scroll through the report to find a listing of the singular values and % variance. Singular value decomposition is an alternative method of solving for eigenvectors (see Chapter 7 of the textbook). Singular values are not eigenvalues, but the axes and the proportions of variance explained by each axis are the same as you would obtain from eigenanalysis of the matrix.

You can save a file of aligned coordinates, centroid sizes, principal component scores and even the consensus plus several others (e.g. bending-energy matrix, eigenvectors, principal warps, uniform component). What you can save partly depends on what window you are in. To save the consensus, go to the **Consensus** display window and go to the **File** menu and select **Save**. You can save the consensus and the aligned shapes. To save the principal component scores, go to the **Relative Warps Display** window, then to the **File** menu and select **Save scores**. Those scores can also be saved by going to the main window, then to the **File** menu and selecting **Save**. From there, you can also save the aligned coordinates (in tps or NTS format) and centroid sizes (in NTS format), as well as the relative warp scores and several other matrices.

MorphoJ (2- and 3D)

In **MorphoJ**, analyses are performed the same way for 2D and 3D data sets. You specify whether your data are 2- or 3D when you load the file. As discussed in Chapter 5, you can also load a wireframe, if you choose, for both 2- and 3D data. The biggest differences between the analyses of 2- and 3D data are the displays of shape differences: 3D shapes can be viewed from different perspectives, but their variation cannot be illustrated by deformed grids.

Principal components analysis

In preceding chapters, we described how to load data into **MorphoJ** and perform a Procrustes superimposition. If you are doing a PCA, you may want to load another file that assigns each specimen to a group so that you can visualize variation among groups. If you are doing a CVA, you will need that file, a group list that **MorphoJ** calls a classifier. Select the data set in the **Project Tree**. In the **File** menu, select **Import Classifier Variables**. Make sure **Match by Identifier** and **The first line of the file contains the names of the new variables** are selected (both are defaults). Find and select the file containing the group identifiers. To use the same files in **MorphoJ** that were used to demonstrate **PCAGen7a**, load *weslm mj.txt* and *wes groups mj.txt*, and click **Open** (if you did not notice - these files have “mj” in their names, meaning that they are formatted for **MorphoJ**). The **Reports** tab will indicate only the number of lines read, so to make sure that the codes were correctly associated with the cases, go the **Preliminaries** menu and select **Edit Classifiers**. The column labeled **Identifiers** should have the IDs in the first column and the second column should contain the group codes (**grp** in *wes groups mj.txt*) and codes should be listed in the correct order. In the case of our sample data set, we can easily check that the codes correctly switch at the right IDs because the group code is embedded in the IDs. Click **Cancel** to exit this screen without making changes.

To perform the analysis, return to the **Preliminaries** menu and select **Generate Covariance Matrix**. Because we only have one data set entered so far, that data set and Procrustes coordinates are already selected, so you just need to click on **Execute**. The covariance matrix will now be listed in the project tree. Go to the **Variation** menu and select **Principal Component Analysis**. The **Graphics** tab will be opened to a new screen showing vectors of landmark displacement for PC1. Click the **Eigenvalues** tab to see a scree plot of % variance.

Click the **PC scores** tab to see a scatter plot of scores on PC1 and PC2. To label by groups choose **Color the Data Points**. (Checking **Label Data Points** will write the identifiers on the plot next to the dots.) In the pop-up window, check **Use a classifier variable…** (in this case, there will be only one variable to choose). To change a default color, select a class and the color you prefer, then click **Use**

Color. When done, click **OK**. (Refer to Chapter 5 for a more extensive discussion of options to edit this graph.)

The eigenvalues and % variance also are tabulated under the **Results** tab; they can be copied and pasted into documents or spreadsheets. Further down in the report listing are the PC coefficients, which are the vectors of landmark displacements.

To export the PC scores, go to **Project Tree** and select the line **PC scores, CovMatrix ….** Go to the **File** menu and select **Export Dataset**. In the window that appears, click on **PC scores** under data type, navigate to the folder where you want the file to be put, and enter a file name.

Canonical Variates Analysis

After entering the data as you did for PCA, both coordinates and a classifier file, go to the **Comparison** menu and select **Canonical Variate Analysis**. In the window that opens, select the data set, data type and classifier. You do have to make a selection even if there is only one choice. If you are using the project opened earlier for PCA, be sure to use the Procrustes coordinates, not the PC scores. To obtain permutation tests of the differences between means, check the box and enter the number of permutations (you might want to start with a smaller number than 10,000). Click **Execute**.

On the **Graphics** tab, you will see **CV shape changes**, and a color-coded scatterplot of scores on CV1 and CV2, which appears in the **CV scores** tab. The deformations shown under **CV shape changes** are computed by regression of shape onto CV scores, and are equivalent to the deformations shown by selecting the **Regr?** option in **CVAGen**.

The listing under the **Results** tab includes the eigenvalues and the proportions of between-group variation they represent, the Mahalanobis and Procrustes distances between group means, and the canonical coefficients (loadings of landmark coordinates on the CV axes), which could be used to reconstruct the deformation along the vector in the CV space. If the permutation tests were selected, those results are shown below the corresponding distances.

Testing the significance of group differences

A test of the null hypothesis that the groups do not differ is not done in **MorphoJ**. Nor are classification or assignment tests performed on CV scores. Rather, **MorphoJ** does pairwise tests using **Discriminant Functions** in the **Comparison** menu. The discriminant function analysis tests the difference between each pair of groups separately, using only the data for that pair. The pooled within-group covariance matrices used in these tests are likely to differ between pairs, and to differ from that used in CVA. Accordingly, the Mahalanobis distances and *p*-values are also expected to differ from those reported for CVA. This will not have a large influence on conclusions if the within-group covariance matrices are similar.

The **Discriminant Functions** analysis includes options to perform all pairwise comparisons, and to perform permutation tests on the differences between selected pairs. In addition to these tests, the **Results** tab also reports classification tables (numbers correctly and incorrectly classified) based on the discriminant function and on a cross-validation test based on a jackknife (leave one out) protocol. Results of the classification test are shown as histograms in the **Discriminant scores** and **Cross-validation scores** tabs for each comparison. If no individuals are misclassified, the plot histograms do not overlap and do not cross the 0 point between the two means. If there are misclassified individuals they will have the "wrong" sign. Misclassified cases may also overlap the other group, although that need not be the case if the within group variances differ greatly.

To identify individual cases that are misclassified, export the appropriate file by going to the **File** menu and selecting **Export dataset**. In the window that opens, select the type of scores you want to export (discriminant or cross-validation), choose the classifier, and make sure the variable names are included in the file. Then select location and enter a filename, as usual. Each column will have the scores for all relevant individuals on that function; excluded individuals will have NaN entered. Misclassified individuals will have the opposite sign of the other members of that group.

R

Principal components analysis

Several functions in **R** will do a PCA. One that is frequently used is **princomp()**, but this often will not work for shape data because the number of variables often exceeds the number of specimens. If that is the case for your data, you will get the error message:

```
Error in princomp.default(filename) : "princomp" can only be used
with more units than variables
```

As a general rule, use **prcomp()** instead because, to our knowledge, this will always work.

If your data are in matrix **Y**, in the standard *x*1, *y*1, *x*2, *y*2,…*x*k, *y*k format (or, if 3D, with the *z*-coordinate for each landmark following the *y*-coordinate) without centroid size in the last column, you can do a PCA by entering:

```
shapePC=prcomp(Y)
```

We assigned the output to **shapePC** so we will find the scores in **shapePC$x**, the standard deviation of the components (the square root of the eigenvalues) in **shapePC$sdev** and the eigenvectors in **shapePC$rotation**. If you want to know the % variance explained by each component and the cumulative % variance explained, enter:

```
summary(prcomp(Y))
```

To get a quick plot of the scores, on the first two PCs, enter:

```
plot(shapePC$x[,1:2], asp=1, pch=21, cex=2.5, col="black")
```

In the last chapter, we discussed how to produce a plot that is color coded by values on a factor (such as **species**) and has a legend as well. To do that, you need a factor as well as the shape data. To reduce the risk of mistakes, it's best to make that factor a column in your data frame that you can assign when you load the file. For example, in the file that contains the shape data for the adult piranhas (**pirAdults**), the first column is the code for the species. So, after loading the file, you would enter:

```
species <- as.factor(Y, [,1])
shape <- as.matrix(Y[,-1])
```

The script for color coding the species uses that factor when filling the symbols in the plot. In the last chapter, we covered the graphical details of this code. What we point out now is that what you plot are the scores, which are in the output ***$x**.

```
palette(c("black","purple","dark magenta", "magenta",
"palevioletred", "sienna", "gold", "goldenrod4", "medium
violetred"))
plot(shapePC$x[,1:2], asp=1, pch=21, cex=2.5,
col="black",bg=(species))
legend('topright',legend=c('dent','el','gould','man','spilo',
'car','nat','pir','alt'),fill=c('black','purple','dark
magenta','magenta','palevioletred','sienna','gold','goldenrod','medium
violetred'),cex=1)
title("PCA piranha adults",cex.main=1.5)
```

To show the deformation along PC1 (or any other), we will usually show the change away from the mean shape towards one extreme. You can pick a value on the *x*-axis (for PC1) that is near to the extreme, e.g. 0.1 for the PCA we show in Figure W6.1. We then multiply that value of 0.1 by the coefficients for the landmarks on PC1, and add that product to the mean shape, converting the result into the **shapes** format. To get the mean shape, enter:

```
meanshape<- t(matrix(colMeans(Y),m,k))
```

The second step is to get the extreme shape. We will break this into two steps, the first produces the quantity that we will add to the mean shape. Here we are using a score of 0.1 along PC1. The second step is to add that to the mean shape.

```
xmax=t(matrix(.1*shapePC$rotation[,1],m,k))
ext=meanshape+xmax
```

We can now plot the difference between the mean and that score. Here we plot it, with the line joining the landmarks (shown in Chapter 5, Figure W5.11B).

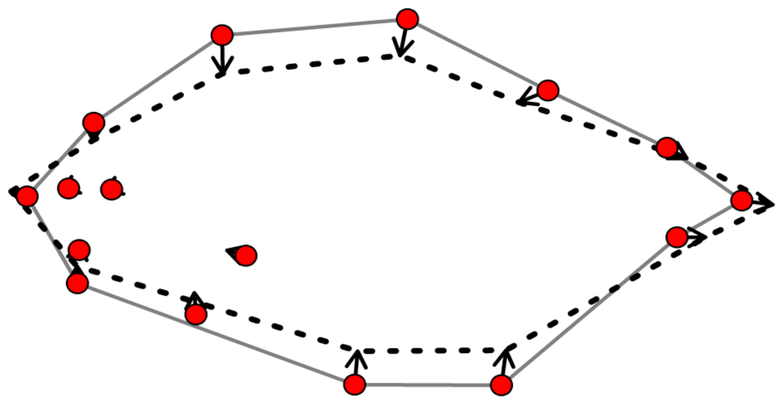


Figure W6.1. The first principal component of variation of adult piranha body shape.

```
plot(meanshape,asp=1,xlab="",ylab="",pch=21,col="black",
bg="red",cex=1,axes=F)
joinline=c(1:10,12,1)
lines(meanshape[joinline,],col="gray50",lwd=2)
lines(ext[joinline,],col="black",lty=3,lwd=3)
arrows(meanshape[,1],meanshape[,2],ext[,1],ext[,2],length=0.1,lwd
=2)
points(meanshape,pch=21,col="black",bg="red",cex=1.5)
title(line=1, "PCA piranha adults", cex.main=1.5)
title(line=2,sub="PC1",cex.sub=1.5)
```

between-group PCA

To do the between-group PCA (bgPCA), we will again use a file in standard format. We begin by centering the coordinates and compute the means of the groups (here, species). We then do the PCA of the means and project the data for all individuals onto the PCs of the means. This could, no doubt, be done more efficiently (and without loops) but this is very easy to follow.

```
#Between-group PCA
Y=as.matrix(Y)
n=nrow(Y)
p=ncol(Y)
k=p/2
#compute grand mean
grandMean=t(colMeans(Y))

#Subtract that grandmean from each row
Y_centered=NULL
  for (j in 1:n){
  centered.temp=(Y[j,]-grandMean)
  Y_centered=as.matrix(rbind(Y_centered,centered.temp))
}

#Calculate the group means
species_means<-NULL
for (i in 1:k){
  mean.temp<-tapply(Y_centered[,i],species,mean)
  sex_means<-cbind(sex_means,mean.temp)
}
#Get scores for all individuals on the eigenvectors of the PCA of means
B=prcomp(species_means)$rotation #eigenvectors
E=Y_centered%*%B
```

We would use the same script for plotting the bgPCs as the PCs, except that here we called the file that contains the scores E. So the first line (excluding the palette) is:

```
plot(E[,1:2], asp=1, pch=21, cex=2.5, col="black", bg=(species))
title("bgPC piranha adults",cex.main=1.5)
legend('topright',legend=c('dent','el','gould','man','spilo','car','nat',
'pir','alt'),fill=c('black','purple','dark magenta','magenta',
'palevioletred','sienna','gold','goldenrod','medium violetred'),cex=1)
```

The script for producing the picture of the deformation is also virtually identical to that for the PCA except that scores and eigenvectors were assigned to **B**.

```
meanshape=t(matrix(colMeans(Y),m,k)
xmax=t(matrix(.1*B$rotation[,1],m,k))
ext=meanshape+xmax
```

Canonical variates analysis/Discriminant functions

A canonical variates analysis can be done using a function in the shapes package, **shapes.cva()** and **lda()** in the MASS package (**shapes.cva()** uses **lda()**) The arguments for the **shapes.cva()** function include the input data (in **shapes** format), a group factor, and whether a Procrustes superimposition should be done (TRUE or FALSE) plus the number of CVs to display. There is a useful function that helps to prepare both the data file and the group factor, **groupstack()**. This function combines the data for up to eight groups into a single file and creates a group factor. **shapes.cva()** presumes that all the coordinates are landmarks (and adjusts the degrees of freedom accordingly), which means that it will treat semilandmarks as if they are landmarks.

To do a CVA for a data file X, with a group factor "groups", you would enter:

```
shapes.cva(X,groups,scale=TRUE,ncv=2)
```

To run **groupstack()**, and then do a CVA on the resultant data, you enter:

```
data <-groupstack(A1,A2…A8)
```

The output of this function is the combined data for all the groups (data$x) and the group labels (data$groups). In the help example for **shapes.cva()**, this function is used to combine data for males and females of two apes, which are then subject to a CVA:

```
data(pongof.dat)
data(pongom.dat)
data(panm.dat)
data(panf.dat)

apes <- groupstack(pongof.dat,pongom.dat,panm.dat,panf.dat)
shapes.cva( apes$x, apes$groups)
```

The function returns a plot of the CVs (if you ask for two or three, i.e. ncv=2 or 3), and it also returns the CV scores.

You can also do a CVA using the **lda()** function in the MASS package. This is useful if you have semilandmarks or many more shape variables than individuals in your file because you can do a PC reduction first and run **lda()** on the PCs. To do the PCA reduction, first read in your data, in standard *x*1, *y*1, *x*2, *y*2,…*xk*, *yk* format. This is your input data matrix Y. To do the PC reduction, replace K with the desired number of PCs in the line below and enter:

```
Z=as.matrix(prcomp(Y)$x[,1:K])
```

Then you do your CVA on matrix Z. You can enter a model formula and the name of the data frame that contains the data, or you can enter the name of the data file and the grouping variable. Here we use the second approach on the reduced data from above:

```
ldaZ=lda(Z,as.factor(group))
```

The function returns: **ldaZ$prior** (the prior probabilities used), **ldaZ$means** (the group means), **ldaZ$scaling**, a matrix that transforms the observations by the canonical variates (the function that circularizes the within-group covariance matrix), **ldaZ$svd**, which are the singular values that produce the ratio of the between to within-group standard deviations on the CVs, and **ldaZ$N** (your sample size). If you ask for a cross-validation test (CV=TRUE) in the argument for the function, you will also get the classification function and posterior probabilities.

To get the Mahalanobis distances, you would compute the Euclidean distances between the group means that are projected into the rescaled space of the CVs.

```
meangroup<-ldaZ$mean
meanproj<-meangroup%*%ldaZ$scaling
dist(meanproj)
```

To get the plot of the scores, you project the data (not just the means in this case) into the rescaled space and then use **plot()** to produce the picture.

```
proj1<-Z%*%ldaZ$scaling
code1=as.numeric(group)
#1 control, 2 liquid, 3 powder
palette(c("gray50","gray80","gray70"))
plot(proj1,asp=1,pch=21, bg=(code1),col='black',cex=2.5)
legend('topleft',legend=c('Group1','Group2','Group3'),
fill=c('gray50','gray80','gray70'),cex=1.2)
```

.

7

Partial Least Squares

PLSMaker7 and **PLS3D7** (IMP), **tpsPLS** (tps) and **MorphoJ** do a Two-Block PLS. All can analyze two blocks of shape data or one block of shape data plus a block of non-shape data (e.g. ecological variables). **PLSMaker** and **tpsPLS** analyze 2D data only; **PLS3D7** analyzes 3D data only and **MorphoJ** analyzes both 2- and 3D data. All the programs can analyze two blocks of shape data as separate configurations, meaning that each configuration is separately superimposed. **MorphoJ** offers a within-configuration approach, in which a configuration is superimposed, then divided into two blocks without subsequently superimposing each block. All the programs will produce the following numerical results: (1) the covariance explained by each axis, which are variously called "SVD" axes ("Singular axis decomposition axes", IMP), PLS axes (MorphoJ), "paired axes" (tps) and, in the literature, singular warps or singular axes (SA); (2) the correlations between scores on each axis for Block 1 and Block 2; and (3) results of permutation tests to determine whether the covariance explained, and the correlation between axes, exceeds that expected by chance. All three programs produce the following graphical results: (1) shape changes along each axis; (2) loadings of non-shape variables on each axis; (3) a scatterplot of the scores for individuals on each axis of the first block versus the second block. In addition to this standard output, **MorphoJ** computes Escoufier's RV and tests its statistical significance, and shows scores for each block on pairs of axes for that block. **PLS3D7** gives the trace correlation.

Dividing the data file into blocks

To analyze two blocks of shape data you need to subdivide a configuration into two subsets. That can be done within **MorphoJ** but, for the other programs, it needs to be done before loading the data into the **PLS** programs. If your files are in IMP format, you can use IMP programs **Lmedit7** or **Vis_Proto7** to create the subsets. To use **Lmedit,** you first need a protocol that lists the landmarks belonging to a block. This protocol consists of two columns, the first of which is the number of the landmark in the **output** file, the second of which is the number of that landmark in the **input** file. So, if there will be *k* landmarks in the subset, the first column contains numbers from 1 to *k*. For example, if the first landmark in the output subset corresponds to the seventh in the input file, and the second landmark in the output subset corresponds to the ninth landmark in the input file, first two lines of the protocol would be:

```
1 7
2 9
```

Vis_Proto7 is a "visual protocol maker" and it is much easier to use than **Lmedit** when you have a large number of landmarks in an order that is difficult to recall correctly. You **load** your data file, go to the **Additional Editors** menu on the tool bar and select **Subset Generator**. In the pop-up window, there is a box where you can type in the numbers of the landmarks that you want to include in the subset. To check that you have written out the correct subset, select **Show Subset** (the landmarks that belong to the subset are then shown by large black dots). If the list you typed is correct, you then **Save Subset**. To produce the second block, you then **Invert this Subset** and save that one by selecting **Save Subset.** You can save the list of landmarks for either block in a text file and copy it into **Vis_Proto** the next time that you need to partition a data file.

If your data are in *.tps format, you would use **tpsUtil** to produce two files. To produce them, on the **Operations Menu**, select **Delete/Reorder landmarks**. Then load the **Input** data file, name the **Output** data file, and go to **Set Up**. Choose the landmarks to include in the first subset and delete the others and save that file. Repeat the process for the other subset. It may be a good idea to write down the landmarks that you excluded from the first block so that you do not inadvertently include some landmarks in both blocks or leave some out of both.

In **MorphoJ**, go to the **Preliminaries** menu and choose **Select Landmarks**. You enter a name for the block and select the landmarks to exclude from it. When you are done, select **Accept**. **MorphoJ** will then open a window for superimposing that block and can align it on it principal axes or on two landmarks. You repeat that process for the second block. When subdividing data files in **MorphoJ**, as in **tpsUtil**, it is a good idea to write down the landmarks that you excluded from the first block so that you do not inadvertently include some landmarks in both blocks or leave some out of both.

PLSMaker7 (2D)

To load the data, click on the **Load Data** button. The first block must consist of shape data (in IMP format). The second block can consist of either non-shape data or shape data. The non-shape data should also be organized with each column containing the data for one variable and the rows containing the data for an individual. If you are loading non-shape data, click on the radio button next to **Landmark Data** for the second block, which will turn off the default. You will notice that there is a third box to allow for loading a third block of data but this option is not yet enabled (check for upgrades). When the data are loaded, they will appear in the visualization window to the left. You can see those plots again by clicking on the **Show Data** buttons located within the field for each block below **Load Data**.

To perform a two-block PLS, click on the **2Block SVD** button below the **Load Data** fields. The numerical results will appear in the orange field at the bottom, although only the results for the first singular value and axis will be initially displayed. To look at those for the second (and subsequent) axes, move **Up** or **Down** the **Active SVD Axes**. You will see the singular value (**SVD score**), the % covariance explained (**SVD percentage)** and the **correlation** between blocks explained by that axis. To determine which, if any, of the singular values and correlations are significant, use the **Statistics** pull-down menu on the toolbar. At present, there is only one option (**Permutation test**). The default is to do 100 permutations, so if you wish to do more, type in the number in the box under **# of Permutations** (located in the purple field of the display options). The results will appear in the **Auxiliary Window**. The first results state the singular value, and the number of times in which a value that high or higher was obtained in the chosen number of random permutations; the final column is the *p*-value for the null hypothesis (that this frequency can be explained by chance). The second set of results, printed below, gives the correlations between the scores of the first and second blocks for each singular axis, and the number of times in which a value that high or higher was obtained in the chosen number of random permutations; again, the final column is the *p*-value for the null hypothesis. It is entirely possible that the singular value is not significant but the correlation is. This occurs when the axis explains a trivial part of the covariance. The results seen in the **Auxiliary Window** can be copied from the window, saved to a new file or appended to an existing file, which is done by clicking on the option **Append Results to File**. (You can safely ignore the caution about overwriting the file because it does not overwrite anything.)

To visualize the relationship between the scores of Block 1 and Block 2, click on **Show Scores** just below **2Block SVD** (both are below the **Load Data** fields). If both data sets are blocks of shape data, the plot can be copied to the clipboard but you will need to use the **auxiliary copy** function; the function that preserves the aspect ratio in plots of the shape transformations interferes with copying the plots of the scores. Alternatively, you can save the scores to files and use the plotting options in a spreadsheet or other program by going to the **File** menu on the toolbar up top and selecting **Save Scores for Block 1** then **Save Scores for Block 2**.

To depict the singular axes as shape deformations (for landmark data) or as loadings of the non-shape variables, click on **Plot Axis** (located in the field for each block, beneath the **Load Data** and **Plot**

Data options). You have the usual options for displaying the shape transformations; some are in the purple field below the visualization window, the remainder are listed in the **Image** pull-down menu on the toolbar up top. In the purple field, you may select **Plot Style**, the **Superimposition** method to use when depicting the deformation. If you select either Bookstein Coordinates (BC) or Sliding Baseline Registration (SBR), make sure you put in reasonable baseline endpoints in the boxes provided on the right side of the purple field. The graphical options, such as the scaling factor, range, density, grid trimming and rotation of the reference are as described in Chapter 5.

You can save the scores for each SA for each block, and the Singular Value Decomposition information (the singular values [S-Value], the percentage of the covariance between blocks explained, [Percentage], and the **U** and **V** matrices). The files of scores are ordered so that SA1 is in the leftmost column, SA2 to the right of it, etc.

3DPLS7 (3D)

To load the data, click on the **Load Data** button. The first block must consist of shape data (in IMP format). The **Load Wireframe** button is to the right of the **Load Data** button. The method for constructing a wireframe is explained in Chapter 5. The second block can consist of either non-shape data or shape data. The non-shape data should also be organized with each column containing the data for one variable and the rows containing the data for an individual. If you are loading non-shape data, click on the radio button next to **Landmark Data** for the second block, which will turn off the default. When the data are loaded, they will appear in the visualization window to the left. You can see those plots again by clicking on the **Show Data** buttons located within the field for each block below **Load Data**.

To perform a two-block PLS, click on the **2Block SVD** button below the **Load Data** fields. The numerical results will appear in the orange field at the bottom, although only the results for the first singular value and axis will be initially displayed. To look at those for the second (and subsequent) axes, move **Up** or **Down** the **Active SVD Axes**. You will see the singular value (**SVD score**), the % covariance explained (**SVD percentage)** and the **correlation** between blocks explained by that axis. To determine which, if any, of the singular values and correlations are significant, use the **Statistics** pull-down menu on the toolbar. At present, there is only one option (**Permutation test**). The default is to do 100 permutations, so if you wish to do more, type in the number in the box under # **of Permutations** (located in the purple field of the display options). The results will appear in the **Auxiliary Window**. The first results state the singular value, and the number of times in which a value that high or higher was obtained in the chosen number of random permutations; the final column is the *p*-value for the null hypothesis (that this frequency can be explained by chance). The second set of results, printed below, gives the correlations between the scores of the first and second blocks for each singular axis, and the number of times in which a value that high or higher was obtained in the chosen number of random permutations; again, the final column is the *p*-value for the null hypothesis. It is entirely possible that the singular value is not significant but the correlation is. This occurs when the axis explains a trivial part of the covariance. The results seen in the **Auxiliary Window** can be copied from the window, saved to a new file or appended to an existing file (which is done by clicking on the option **Append Results to File**. (You can safely ignore the caution about overwriting the file because it does not overwrite anything.)

The trace correlation can be computed by clicking on the **Trace Correlation** button; when sample sizes are small, you should use the **Generalized Inverse**; the results will appear in the **Auxiliary Window**. Because the trace correlation is a measure of the covariance explained by one block, it matters whether Block 1 or Block 2 is predicting the other. The results are shown for both cases, i.e. with Block 1 predicting Block 2 and also for Block 2 predicting Block 1.

To visualize the relationship between the scores of Block 1 and Block 2, click on **Show Scores** just below **2Block SVD** (both are below the **Load Data** fields). If both data sets are blocks of shape data, the plot can be copied to the clipboard but you will need to use the **auxiliary copy** function; the function that preserves the aspect ratio in plots of the shape transformations interferes with copying the plots of the scores. Alternatively, you can save the scores to files and use the plotting options in a spreadsheet or other program by going to the **File** menu on the toolbar up top and selecting **Save Scores for Block 1** then **Save Scores for Block 2**.

To depict the singular axes as shape deformations (for landmark data) or as loadings of the non-shape variables, click on **Plot Axis** (located in the field for each block, beneath the **Load Data** and **Plot Data** options). **The Deformation Display Formats** are to show the wireframe on the reference, target or both, and you can also show the deformation by vectors at landmarks (the option that will be used automatically if you do not load a wireframe). The graphical options, such as the scaling factor, range, density, grid trimming and rotation of the reference are as described in Chapter 5.

You can save the scores for each SA for each block, and the Singular Value Decomposition information (the singular values [S-Value], the percentage of the covariance between blocks explained, [Percentage], and the **U** and **V** matrices). The files of scores are ordered so that SA1 is in the leftmost column, SA2 to the right of it, etc.

tpsPLS (2D)

Before loading any data, you need to specify the **Mode**, i.e. to say whether you will analyze the relationships between shape and non-shape variables or between two blocks of shape variables. The default mode is **Variable** (non-shape)/**Shape** mode, so if both your blocks are shape variables, click on the radio button to switch to **Shape/Shape** mode. The non-shape data must be in NTS format, the shape data can be in either tps or NTS format. Now load the data. If one block consists of non-shape data, that block is entered as the **Variables** file. Then the shape data are loaded as the **Shape** file. As each button is enabled, click it to produce first the **Consensus**, then the **Partial warps**, then **PLS** and, if desired, the **Permutation** test.

To see the numerical results, go to the **File** menu and select **View Report**. Look for the section titled ***Cross set covariance***; this gives the proportion of the total covariance explained by each dimension (SA). Below that you will find the ***Correlations between variable and shape vectors***. At the end of the report listing, you will find the results of the permutation tests. You are looking at the number of permutations (**Count**) and percentage of them (**Percent**) that are as large as or larger than the observed values. The first one given is the proportion of the covariance explained by each dimension, the second is the correlation between the paired dimensions (i.e. SA1 of the first block and SA1 of the second). So, if percentages are lower than 5%, that means that the observed value exceeds what is expected by chance. To do the graphics, go to the Visualization window. You can visualize the shape changes along each axis, and the scatterplots for the scores of Block 1 and Block 2 on each singular axis; the graphical options are as described in Chapter 5.

MorphoJ (2- and 3D)

MorphoJ does both a **Two-Separate Blocks** and a **Blocks Within One Configuration** PLS. We describe these separately although both are accessed from the **Covariation** menu, **PLS** submenu.

Two-Separate Blocks PLS

When you have assigned the landmarks to blocks, go to the **Covariation** menu, select **Partial Least Squares**, then **Two Separate Blocks**. In the pop-up window, at the top, there is a box where you can name the analysis. Below that are two boxes, one on the right and one on the left. Select the first block, then in the window beneath that, select the data matrices (e.g. Procrustes coordinates) and, in the window below that, select the variables (e.g. Procrustes coordinates). Then do the same for the second block. Select both the sets in the bottom window (both should be highlighted when both are selected). If you want to do a pooled-within subgroup analysis, check that box. What a pooled-within subgroup analysis does is remove the variation among the subgroup means from the analyses; the analyses are conducted on the deviations from those means. If you select this, you also need to select the classifier. If you want the permutation tests to be done, check that box as well. When you are done, click **Accept**.

The numerical results consist of the coefficients for the singular axes, the singular values for each axis, the covariance between the two blocks along each axis, the correlation between the two blocks along each axis, and the RV coefficient. If the permutation test option was selected, the results also include the permutation P. The graphical results include a bar chart of the amount of covariation explained by each singular axis, the shape changes along each singular axis for each block, scatterplots of the scores for Block 1 versus Block 2 on each axis, and for pairs of singular axes within each block. If you selected the pooled within-subgroup analysis, you will also see a scatterplot of the group-centered scores (the average score for each group is 0.0, 0.0). The options for the graphical outputs are as described in Chapter 5.

Two-Blocks Within One Configuration PLS

To do **Two-BlockPLS within-configuration**, go to the **Covariation** menu, select **Partial Least Squares**, then **Within a configuration**. You then need to select the landmarks that belong to each block; a large window will open that shows your landmarks (numbered) and you select the ones that belong to Block 1 or Block 2. Because all are initially assigned to Block 1 and colored red, it is easiest to select those that belong to Block 2 and they will be colored blue. When you are satisfied with your subdivision, select **Accept**. A very useful function is the ability to copy the block design from one data set to another. To do that, select **Copy the blocks of landmarks**, then go to **Select subsets of landmarks** and paste it into another data set by selecting **Paste Partition**. When you have selected the landmarks, click **Accept** and the **Start Analysis** window will open. Give the analysis a name, check the **Permutation** test, if desired, and the **pooled within-subgroups analysis**, if desired. If you want the pooled within-subgroups analysis, you also need to select the classifier for the groups. Then click **Accept**.

The numerical results consist of the coefficients for the singular axes, the singular values for each axis, the covariance between the two blocks along each axis, the correlation between the two blocks along each axis, and the RV coefficient. If the permutation test option was selected, the results also include the permutation P. The graphical results include a bar chart of the amount of covariation explained by each singular axis, the shape changes along each singular axis for each block, scatter plots of the scores for Block 1 versus Block 2 on each axis, and for pairs of singular axes within each block. If you selected the pooled within-subgroup analysis, you will also see a scatterplot of the group-centered scores (the average score for each group is 0.0, 0.0). The options for the graphical outputs are as described in Chapter 5.

8

Statistics

This chapter covers implementing simple statistical analyses, by which we mean linear regression of shape on a single independent variable, comparison of two mean shapes, or one-way multivariate analysis of variance (MANOVA). If you have multiple means to compare, you would presumably do the MANOVA first, followed by pairwise comparisons if the null hypothesis of equal means is rejected. For that reason, this chapter covers MANOVA before pairwise comparisons. If you have just two groups to compare, you can go directly to the final section. Although you can use a MANOVA to compare just two means, there are programs that compare only two. Those are in the final section of this chapter.

Rather than organize this chapter around programs, presenting each one and everything that it does, we have organized this chapter around procedures. That means that you can locate the procedure that you want to do, find the program(s) that will do it and read how to implement the analysis using each program. However, for readers who want an overview of what procedures are available in each comprehensive package, and which ones do what, we begin this chapter with a brief synopsis, organized by package.

Synopsis: What the programs do

- **IMP series**
    - *Regression*: **Regress7a, Regress7** (both 2D) and **ThreeDRegress7** (3D) do a regression. **Regress7a** gives the variance explained and a permutation test of the statistical significance of the variance explained. **Regress7** also gives Goodall's *F*, tested by permutations. **ThreeDRegress7** gives the results of a multivariate test (Wilk's Lambda), and the permutation test of the variance explained.
    - *MANOVA***: CVAGen** does a MANOVA, giving the results of both a multivariate test of the equality of the means (Wilk's Lambda) and a permutation test.

  - *Pairwise comparisons:* **Two-Group** (2D) performs pairwise comparisons by Goodall's *F*-test; giving both a parametric *p*-value and a permutation *p*-value and the Procrustes distance between the means. **Simple3D** does a pairwise comparison for 3D data using Goodall's *F*-test, tested by permutations.

- **tps series**
  - *Regression*: **tpsRegr** does regression. **tpsRegr** gives the results of a multivariate tests of the null hypothesis if the sample size is large enough, Goodall's *F*-ratio and parametric *p*-value and, if requested, a permutation *p*-value plus the variance not explained by the independent variable.
  - *MANOVA***: tpsRegr** does a MANOVA giving the output described above for regression.
  - *Pairwise comparisons:* **tpsRegr** will do pairwise comparisons between means.
- **MorphoJ**
  - *Regression*: **MorphoJ** does a regression; the procedure is located in the **Covariance** module.
  - *Pairwise comparisons:* **MorphoJ** does pairwise comparisons as part of the **Comparison: Canonical Variates Analysis** or **Discriminant Functions** modules. The results include Procrustes and Mahalanobis' distance between means; Hotelling's T^2 and parametric *p*-value and, if requested, permutation *p*-values.
- **PAST**
  - *MANOVA*: **PAST** has two options for MANOVA, both found on its **Multivar** menu. **MANOVA/CVA** does the multivariate tests if sample sizes are large enough; the results include Wilk's Lambda, Pillai's trace and parametric *p*-values. The second is **One Way NPMANOVA,** a permutational MANOVA (on the pairwise distances between individuals). The results are an *F*-ratio and a permutational *p*-value. The results include tests of all pairwise differences between means, with Bonferroni or sequential Bonferroni correction as options.
  - *Pairwise comparisons*: **PAST** also has a procedure for a pairwise comparison **Two-group permutation test**, also on the **Multivariate** menu. This gives both a Mahalanobis' distance between means and a permutation *p*-value when sample sizes are large enough.
- **R**
  - *Regression*: All statistical analyses can be done in **R**. Regression can be done using the **lm()** function and a permutational test (on pairwise distances between individuals) can be done in **adonis** (in the **vegan** package).
  - *MANOVA*: MANOVA can be done in **R** using the same functions as regression.

- *Pairwise* comparisons: The **shapes** package includes two functions that do pairwise comparisons between means, **resample** and **testmeanshapes**. Regression, MANOVA and comparisons between means can be done using programs that are not part of the shapes package: **lm()**, with the *F*-ratio adjusted for the appropriate degrees of freedom, and **adonis()**, which does a permutational MANOVA.

Implementing regression

Regression in Regress7A, Regress7 (2D)

Regress7a (and the two other IMP regression programs) regresses shape on the last column of an IMP data file, the centroid size column. If you want to regress on a variable other than centroid size, paste it into that column, replacing centroid size. When you open **Regress7a**, you first need to tell the program whether to regress on **CS or Log(CS)**. It does not matter whether you are regressing on centroid size or some other variable, what matters is whether you want to regress on the value in the last column or on a log-transformed value. In the bright red box below you will see several options that all have to do with the choice of the reference. For purposes of statistical inference, select the top option, **Use GLS reference**. The other choices are occasionally useful, but not for purposes of statistical inference (see below for their purposes). To find the statistical results, go to the **Regression statistics** menu at the top. The first option regresses the Procrustes distance from the reference on size; these results will rarely be of any interest unless you are calibrating the magnitude of the change in shape relative to size. The pop-up window alerts you to the fact that this is not a test of the significance of the regression. The second gives the variance in shape explained by size. The third gives the results of two statistical tests. One is the generalized Goodall's *F*-statistic, the other is the variance explained by the regression; the permutational *p*-values (based on bootstrapping) are shown below the values for the test statistics.

There is one graphical option that is related to the statistical test of the Procrustes distance on size. This is the **Display Distance on CS or LCS**. The plot is useful for two main purposes; first, it gives you a quick visual check on the linearity of the relationship between shape and the independent variable; this is useful when you are debating whether to use centroid size or the log-transform of centroid size. Second, it depicts your sampling scheme; you can see if a few specimens at the smallest or largest sizes have a large influence on the regression. The other plot is the change in shape (**Display Regression (Deformation)**). The default display is **Grid + Vectors**; you can also get a picture of the **Vectors on the Smallest**, which uses the smallest specimen, or the one with the lowest value on another independent variable, as the reference for the plot. Rather than showing the change away from the mean, the picture shows change away from the smallest value. The graphical controls are in the **Display** format on the toolbar (the controls for the axes are in the **Axis Control** menu on the toolbar). If you want to look at the result using a different superimposition, those options are in the lime-green box below the options for the reference. The **trim grid** and **reference rotation action** functions are in the pink **Display** box below the graphics window. You can save the results (the regression statistics) and the regression vector (either normalized to unit length or not). These options, plus the options to save the partial warps, the Procrustes distance of each individual from the reference, and the column of independent variable values, are in the lavender **Save** box below the **Deformation Plot Format** box. They are also above, in the **File** menu on the toolbar.

If you want to show the deformation from the smallest specimen, you can use any of the graphical displays, not just the vectors on the smallest. To do that, after obtaining your statistical results, reload your data and change the reference from **GLS** to **N Smallest as Reference**. If you have only a few individuals with very low values on the independent variable, pick a small number (e.g. 3). You can also load a reference computed from another analysis if you want to save the regression vectors, or partial warp scores to make them comparable to those from another analysis. Use the **Load Reference** box for that.

Regress7 has a very different interface although the same functions are on it as on **Regress7a**. **Regress7** retains the interface of an early version of the program. To run **Regress7**, you need to **Set Superimposition Type** (which affects only the graphics) before you load your data (green box numbered 1), then say whether to **Regress on** the value in the last column or a log transformed value (violet box numbered 2). Then **Load data** (blue box numbered 3), **specify the reference** (red box numbered 4) and **Compute Partial Warps Scores** (green box numbered 5). To get the statistical results, go to **Regression Statistics** on the toolbar and select the desired test. You will not see any image in the graphics window until you ask for it, in the turquoise box (**Display**), **Display Regression (Deformation)**.

Regression in ThreeDRegress7 (3D)

To run **ThreeDRegress7**, load your data file and, if desired, your wireframe**.** You can run the program without a wireframe, but it may be difficult to interpret the plot without one. Within the box **Load File**, right below where you see the name of your data file, select either **Use X = CS** (which means to use the last column in your data file, not necessarily CS) or to **Load Independent Var (X)**, a file that contains the values for the independent variable in a one column text file, with no headers or row labels. In the purple box below that are several functions. The first one, **Specify Function of X**, asks you to say whether you want to regress on the independent variable, as it is, or log-transform it. The next one, **Carry Out Regression**, does the regression. To see the statistical results, go to the **Statistics** menu on the toolbar and select the desired test; if your sample size is less than three times the number of landmarks plus semilandmarks, the multivariate test will return the values NaN. The only alternative is the permutation test of the variance explained. To see the graphical results, select **Display Regression**, which is adjacent to the

Carry Out Regression function. When you select that display, the picture will appear in the pop-up **Figure** window. You can label the landmarks on the figure using the **Add Labels to Plot button**. To save the figure, copy it to the clipboard using the **3D to Clipboard** button in the yellow **Copy 3D Image box**, or copy it to an *.eps file (**3D to EPS**). On the pop-up **Figure** window, the **Save** button offers many additional figure file formats. As well as saving the figure, you can save the regression information by copying it from the **Results Box** below the buttons for the functions that carried out the regression. You can also save the regression vector, partial warp scores and the reference (these options are in the salmon pink **Save** box below the **Figure** window on the main interface).

Regression in tpsRegr

To run **tpsRegr**, you need a shape data file and an independent variable file. The shape data file can be in either *.tps or *.NTS format, the independent variable must be in *.NTS format. The first line of the independent variable file thus contains the control line giving the matrix type (**1**), number of rows (which is your sample size), number of columns (the number of variables in the file), and the missing data option (**0**). If you labeled the independent variable, add an **L** to the third (column) number, e.g. 1L. The remainder of the file consists of the column(s) of values for your independent variable. For studies of allometry, you can produce your independent variable file within **tpsRegr** if you used the scale factor in **tpsDig**. Go to the **File** menu, select **Save** and then select **centroid size**. That produces a file of centroid sizes in *.NTS format. If you want to do the analysis using log-centroid size instead, open your *.NTS file in a spreadsheet and add a second column, this one of log-transformed values. Just remember to change the control line to a **2** for the number of columns and whenever you have two or more variables in the file, it is useful to include their labels, so make sure that the control line has **2L** for the number of columns. Below that control line, add a line with the two labels in it, e.g. **CS LCS** (no punctuation goes in this line). Save that as a text file with an *.NTS extension (you may have to change the name of the file if your spreadsheet automatically adds *.txt to all files that are saved as text files).

Load your data file into **tpsRegr** and then load your independent variable file. If you have more than one independent variable, go to the **Options** menu and then to **Select independent variable**. Pick one of your independent variables (unless you intend to do a multiple regression). There are several other options on this menu that allow you to pick whether the specimens should be scaled to unit centroid size or rho, whether the data should be projected onto the tangent space using an orthogonal or stereographic projection, whether to include the uniform component, and whether to do a multivariate or multiple regression. The defaults are to scale to unit centroid size, use an orthogonal projection, include the uniform component, and do a multivariate regression. Once you have selected the options you want, you start the analysis by clicking on the **Consensus** button, and then, when it becomes enabled, on the **Partial warps** button and then on the **Regression** button and finally on the **Perm tests** button. This last button will open a pop-up window asking how many permutations, and in the **Permute** box, how the data should be permuted (**All**, **Within blocks,** or **Among-Blocks**). If you do not have blocks that should be treated as single units (e.g. multiple replicates of the same individual), then use the default, **All**.

You will find the results by going to the **File** menu and selecting **View report**. There is a great deal of output. At the end of it, you will see the **Generalized Goodall *F*-test**, the **% variance not explained**, and the results of the permutation test, which tell you the percent of the time that the observed value for two test statistics, Wilk's Lambda and Goodall's *F*, is exceeded by the ones obtained by randomly permuting the data. You can scroll up to find the parametric tests. To see the picture, go to **Visualization**. What you will see is a deformed grid that does not look deformed at all. That is because it is showing the change from the mean shape to the mean shape. To see how the mean changes, move the **slider** all the way to the right. That shows the change from the mean to the largest specimen. If you want to see the vectors instead of a grid, go to the **Options** menu on the **Visualization** plot and select **Vectors**. You can save the plot to the clipboard or as an enhanced metafile (*.emf) or a bitmap (*.bmp). If you want to edit the picture in a graphics program, select *.emf. If you try to insert the deformed grid directly into your graphics program, by copying it from the clipboard and pasting it into the graphics program, the graphics program may not read it properly. That can be fixed by pasting the image into another program first (we pasted it into **Word**) and then copying the figure from that file and pasting it into the graphics program. Unlike the plot of the deformed grid, the plot of the vectors seems to copy properly without the intermediate step.

Regression in MorphoJ

If you don't have a project open, or a saved project to open, create one now following the instructions we gave earlier (Chapter 1). If you want to regress shape on a covariate that is not the centroid size computed from the coordinates you loaded, then you also need to load a covariate file containing that independent variable. That file needs the specimen IDs in the first column, which is the same first column as the one in your data file. If you loaded a tps file, the identifiers will be taken from the "ID =" line so the first one will have the value ID = 0. Thus, the first specimen in the covariance file must also be "0". Then label the second column CS, or LCS, or whatever you are using as your independent variable. Just to make sure that there are no mistakes, select your data set in the **Project Tree**, go to the **Preliminaries** menu, select **Edit Covariates File** and check that it looks correct.

To do the regression, first go to the **Project Tree** and select your data set and, if you have not already done so, go to the **Preliminaries** menu and select **New Procrustes fit**. Then go to the **Covariance** menu and select **Regression**. In the pop-up window that appears, you need to select first your **Dependent variables**, then the **Independent variables** from the (many) available options. For the dependent variables, select your **Dataset**, then the data matrix (yourdataset, **Procrustes coordinates)** and then the **Variables** (again, **Procrustes coordinates**). Now, for the independent variable, you again need to select your data set. For data matrices, choose either **centroid sizes** (as estimated by **MorphoJ** from the scale factor, if that was present in your tps file or else from the coordinates that you loaded) or **Covariates** (if you loaded a covariates file). Finally, select the Variables. Make sure that you have actually selected the two data sets; the ones that you choose will appear in the **Variables** window but they are not actually selected unless they are highlighted. Select **Perform permutation test** (and a small number of permutations, to start), then **Execute**. To see the results, go to the **Results** tab, next to the Project Tree tab. Scroll to the bottom to see the results. What you will see is the **%variance predicted** and the **permutation-p value**.

MorphoJ gives two plots. One, **Shape changes**, shows the change in shape with increasing scores on the independent variables. The other is a scatterplot that shows scores for each individual on a shape axis (**Regression score1** axis) relative to their value on the independent variable, on the *x*-axis. This plot is obtained by projecting each individual's data onto the regression vector (just like principal component scores are obtained by projecting each individual's data on the principal components).

Regression in R

Here we cover doing a regression in **R** on data superimposed in **R** using either the **procGPA()** function of the **shapes** package or Claude's **pgpa()** function (Claude, 2008). Using either of these functions involves not only doing the superimposition, but also reformatting the data so that each individual's data occupies a row, the format expected by the statistical programs in R. We closely follow Claude's script for a regression, but we reformat the data differently so that the coordinates are sequenced *x*1, *y*1, *x*2, *y*2, …, *xk*, *yk*. In addition, we use *k* for the number of landmarks and *m* for the number of coordinates. We also modified a line in the script so that the degrees of freedom for the parametric *F*-test will be correctly calculated when there are semilandmarks in the file.

Load your data, using one of the **read.tps** functions, if your data are in tps format, or **importToShapes** if your data are in IMP or **MorphoJ** format. If you use **importToShapes()**, your data, formatted for **shapes**, will be in **res$data**. If, when running **importToShapes(),** you assigned the output to an object named **myData**, the formatted data will be in **myData$data**. Centroid sizes will be in **myData$cs**. Typing is easier if you rename these files. Here we will rename them **myDataShape** and **myDataSize**, but first we will log transform centroid size.

```
myDataSize<-log(myData$cs)
myDataShape<-procGPA(myData$data)
myDataShape<-myDataShape$rotated
n=dim(myDataShape)[3]
m=dim(myDataShape)[2]
k=dim(myDataShape)[1]
```

In order to do the statistical analysis, we have to reformat the data, using the approach implemented by Annat Haber to make it readable by a statistical function in **R**. We do not want to overwrite the file that is formatted for shapes in case we need it later, so we will call our reformatted data **dataShape**.

```
dataShape <-t(apply(myDataShape,3,t))
```

Then we do the regression and assign the output to "model1"

```
model1=lm(dataShape~(log(myDataSize)))
```

Now we are going to calculate F. To do that, we first need to compute the mean shape, which we will do using the apply function. We are going to calculate column means (hence the 2 in argument for the function). We are then going to make as many copies of that mean as we have individuals in the file.

```
meanShape=apply(dataShape,2,mean)
meanShape=rep(1,n)%*%t(meanShape)
```

We now calculate *F* by computing the sum of squared deviations between the predicted values and the mean shape. This, divided by the degrees of freedom, gives the numerator for the *F*-ratio. Then we compute the sum of squared deviations between observed and predicted values which, when divided by the degrees of freedom, gives the denominator of the *F*-ratio. Note that we are not actually dividing by the degrees of freedom for shape now - this does not matter because the ratio between the numerator and denominator (*F*) will not be affected.

```
num<-sum(apply((model1$fitted.values-meanShape)^2,1,sum))/1
den<-sum(apply((dataShape-model1$fitted.values)^2,1,sum))/(n - 1 -1)
```

Now we calculate *F* as the ratio between the numerator and denominator and we show the result, then we determine its parametric p-value. At this point, we need to distinguish landmarks from semilandmarks in computing degrees of freedom; we cannot equate *k* to the number of rows in our shapes-formatted data. Instead, we need to input the values for *k* (number of landmarks) and *l* (number of semilandmarks). For 2D data, we can compute *F* and *p* using:

```
Fs<-num/den
Fs
P<-1-pf(Fs, 1 * ((k*m)+l) -4, (n-1-1)*((k*m)+l)-4)
P
```

If you have 3D data, you will also want to modify the expression for *p* according to whether you have 3D landmarks only or 3D landmarks and semilandmarks.

Now we calculate the variance explained by the regression and show the result:

```
vexp<-sum(diag(var(model1$fitted.values))) /sum(diag(var(dataShape))
vexp
```

We also want to include a permutation test of the *F*-ratio, which we can do using **adonis()** in the **vegan** package (Oksanen et al., 2011). If you did not install **vegan** when you first got **R**, do so now.

```
Results=adonis(dataShape~myDataSize,method="euclidean")
Results
```

This will produce Table 1.

Table 1. Ontogenetic allometry of *S. gouldingi* analyzed by a regression of shape on log centroid size (analyzed in adonis)

Effect	Df	Sums of Squares	Mean Squares	F.Model	R2 Pr(>F)	*p*
log(myDSize)	**1**	**0.105628**	**0.105628**	**94.025**	**0.723**	**0.001 *****
Residuals	**36**	**0.040443**	**0.001123**		**0.27687**	
Total	**37**	**0.146071**			**1.000**	

The next step is to produce the reference and target shapes so that we can make the figures. We are going to use the fitted values, finding the one at the minimum size to be the reference shape and the one at the maximum size to be the target shape. To make typing simpler, we will rename **mod1$fitted.values** as **fit.val**. When we produce the two matrices, we also need to reformat them, putting them back into the shapes format for the purpose of the graphics. Because we redefined *k* above (when we distinguished between landmarks and semilandmarks), we will redefine it again as the number of landmarks plus semilandmarks, which is the number of rows in a matrix in the **shapes** format**.**

```
k=k + l
fit.val<-mod1$fitted.values
dataMax<-t(matrix(fit.val[which(Dsize==(max(Dsize))),],k,m))
dataMin<-matrix(fit.val[which(Dsize==(minDsize))),],k,m)

matr=dataMin
matt=dataMax
```

Now that we have these two shapes, we can use any of the graphical styles discussed in Chapter 5. For example:

```
plot(matr, asp=1, type="n",xlab="", ylab="",axes=F)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
points(matr, pch=21,col="black", bg="red", cex=1)
title(main="Ontogenetic shape change",
line.sub=-1,sub="Serrasalmus gouldingi",font.sub=3)
```

Implementing multivariate analysis of variance (MANOVA)

Multivariate analysis of variance (MANOVA) for 2D data can be done in **CVAGen7b** (IMP), **tpsRegr** (tps), **PAST** and **R**; **PAST** and **R** can also analyze 3D data. If you already ran **CVAGen7b** you need only to find the functions that report the results of the MANOVA. If you did a regression in **tpsRegr**, you only need only to learn how to code the independent variable file for the MANOVA. Similarly, if you did a regression in **R** you only need only to learn how to encode your independent variable as a factor.

CVAGen7b(2D)

When you run **CVAGen7b**, the **Auxiliary Results** window reports the results for the multivariate test of the equality of group means, along with a parametric *p*-value. The first value reported is the *p*-value for the test of the equality of all the means. There is another function in **CVAGen** which does a permutational MANOVA. This function is on the **Statistics** menu, the **Single Factor Permutation MANOVA** (about half-way down the menu). When you select that option, a pop-up window asks how many permutations you want (the default is 200). When it finishes running, you get a MANOVA table (Table 2) with the *p*-value for the null hypothesis of equal means.

Table 2. Permutational MANOVA of adult piranha body shapes from CoordGen, Single Factor Permutational MANOVA on the Statistics menu

Effect	Sum of Squares	DF	MS	*F*
Group	1.354224	8	0.169278	150.765023
Error	0.427784	381	0.001123	
Total	1.782009	389		

***p* value =0.009901 based on 100 permutations**

tpsRegr(2D)

To do a MANOVA in **tpsRegr**, you need to prepare the independent variable file. This is done by coding a design matrix (a topic we discuss in more detail in Chapter 9). One way to do this, before you read the more complete discussion, is by "effect-coding". This uses only 1s, 0s and -1s to code all the members of all the groups. For an analysis of four groups, in the first column, you code all members of the first group **1** and all members of the last group **-1** and all members of the groups between them as **0** (Table 3). In the second column, you would code all the members of the second group **1** and all the members of the

Table 3. A group list for CVAGen, in the first column, and effect-coding of those individuals. If you are preparing the independent variable file for tpsRegr, you would use the last three columns (grp1, grp2, grp3) below the *.NTS control line as your independent variable file)

GroupList (CVAGen)	grp1	grp2	grp3
1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0
2	0	1	0
2	0	1	0
2	0	1	0
2	0	1	0
2	0	1	0
3	0	0	1
3	0	0	1
3	0	0	1
3	0	0	1
3	0	0	1
4	-1	-1	-1
4	-1	-1	-1
4	-1	-1	-1
4	-1	-1	-1
4	-1	-1	-1

last group **-1** and all others (both the first and third group) as **0**. In the third column, you would code the third group **1**, the last **-1** and all others by **0**. For an analysis of four groups, you need only three columns. (You do not need a first column of 1s because the analysis is done on the partial warps so the data are mean-centered). In the next chapter, we introduce a function in **R** that will prepare the design matrix for you (**model.matrix**).

The first line in the independent variable file is the standard *.NTS control line, which for the file coded in Table 3 has 20 rows and three columns; if these are labeled the first line in the file is:

```
1 20 3L 0
The next line reads:
grp1 grp2 grp3
```

After that line are the three columns of codes.

Coded this way, the constant is the grand mean and the coefficient for each group is the difference between its mean and the grand mean.

To do the analysis, load your data file and the independent variable file and, on the **Options** menu, go to **select indep variables** and select **grp1 grp2 grp3**. Then you do the regression, first computing the **Consensus**, then **Partial warps**, then **Regression** and then, if desired, **Permutations**. To see the results, go to the **File** menu and select **View Report**. The output includes the multivariate tests (if sample sizes are large enough), and the generalized Goodall's *F*-test. If permutations are requested, these results are at the end of the file: the number and percentage of *F*s equal or larger than the observed one. To see the graphical results, go to the **Visualization** window.

PAST (2- and 3D)

To enter your data into **PAST**, you can just copy it from a tab-delimited spreadsheet and paste it into the spreadsheet-like data window. To tell **PAST** which individuals belong to which groups, you select the rows, then go to the **Edit** menu, select **Row Color/Symbol** and assign a different color/symbol to each group. Then select all the groups that you wish to include in the analysis and go to the **Multivar** menu on the toolbar. There are two options for the MANOVA. One, called **MANOVA/CVA** on the **Multivar** menu, does a multivariate test, reporting Wilk's Lambda and Pillai's trace and a parametric *p*-value. This option can be used only when the number of individuals exceeds the number of variables. The other, which can be used when the number of variables exceeds the number of individuals, is the **One Way NPMANOVA,** a permutational MANOVA. The results of this procedure are an *F*-ratio and a permutational *p*-value.

To do the parametric MANOVA: select all the groups that you want to include in the analysis (the individuals within groups should be in contiguous rows, and the rows to be included in the analysis should also be contiguous). Go to the **Multivar** menu, select **MANOVA** and, in the pop-up window for the analysis, look for the word **constraints** (it is in the middle of the interface, between the two columns of text on the left and the two on the right). For 2D data, select **4** and for 3D data, select **6**. As well as giving the test of the overall equality of means, this function gives all the pairwise comparisons between means. When testing the statistical significance of the differences, the options are to use the *p*-values uncorrected for multiple tests, the sequential Bonferroni, or the Bonferroni adjustment. There is also an option to get the squared Mahalanobis distances between means. To do the permutational MANOVA, select all the groups that you want to include in the analysis, go to the **Multivar** menu and select **One-way NPMANOVA.** This test uses distance matrices, and the default distance is the Bray-Curtis distance, so when the window opens (after the initial analysis is done) select **Euclidean distance** and rerun the analysis. As well as giving the *F*-ratio and permutational *p*-value for the overall MANOVA, this function also gives the *p*-values for pairwise comparisons, with the same three options for adjusting the *p*-values for multiple comparisons (no adjustment, the sequential Bonferroni adjustment and Bonferroni).

R

If we simply import an IMP file into **R** and do the statistical analysis on those coordinates, the analysis is very simple. If, instead, we want to reformat the data for shapes, do the Procrustes superimposition, then reformat them for the statistical analysis, and then reformat them for the graphics, we have made the analysis more complicated. However, the advantage of the more complicated procedure is that it does not presume that the data in your IMP file were all superimposed at one time. They might have been copied and pasted from several files. So we begin by importing and superimposing the data. But we need to import a second file, the one that contains the factor. If we put that into the centroid size column of our IMP file, it will be in the **$cs** output from **importToShapes()**. We will also read in a file that contains the information about our independent variable (this file has a header, the name of the independent variable).

So we will read in two files:

```
myData<-importToShapes(file.choose())
myFactor=read.table(file.choose(), sep="",header=TRUE)
```

We now need to tell **R** that **myFactor** is a factor

```
myFactor=as.factor(myFactor)
```

Now we use the output of **importToShapes()** in **procGPA()**

```
myDataShape<-procGPA(myData$data)$rotated
n=dim(myDataShape)[3]
m=dim(myDataShape)[2]
k=dim(myDataShape)[1]
g=nlevels(myFactor)
```

We again need the shape data in a format readable by the statistical package, and we also need to compute n, p, k, and g (the number of groups, or, equivalently, the number of levels of our factor)

```
shapeData <-t(apply(myDataShape,3,t))
n=nrow(myDataShape)
p=ncol(myDataShape)
```

As we did before, when regressing shape on size in **R**, we will write out our model and calculate *F* and test its statistical significance and compute the variance explained.

```
model1=lm(shapeData~myFactor)
meanShape=apply(shapeData,2,mean)
meanShape=rep(1,n)%*%t(meanShape)
num<-sum(apply((model1$fitted.values-meanShape)^2,1,sum))/(g-1)
den<-sum(apply((shapeData-model1$fitted.values)^2,1,sum))/(n-1 -1)
Fs<-num/den
Fs
```

As before, we need to separate the landmarks from semilandmarks when computing the degrees of freedom for shape in calculating the *p*-value:

```
#If you have semilandmarks, input k and l in the following lines
and remove the #signs
k = k
l =0


P<-1-pf(Fs, 1 * ((k*m)+l)-4, (n-1-1)*((k*m)+l)-4
P
vexp<- sum(diag(var(model1$fitted.values)))/sum(diag(var(shapeData)))
vexp
```

and we use **adonis()** to do a permutational MANOVA:

```
Results <- adonis(shapeData~myFactor,method="euclidean")
Results
```

Then we do the graphics by computing the mean shapes that we want to compare and reformatting them. If you want to show pairwise differences between means of all the groups, compute the mean shape for each. However, because we discuss pairwise comparisons below, including the graphical output of the comparisons, we do not include that in this section.

Pairwise comparisons of means

Pairwise comparisons between means of 2D data can be done in **Two-Group** (IMP), **tpsRegr**, **MorphoJ**, **PAST**, and **R**. In **PAST**, MANOVA automatically gives the pairwise comparisons. In **MorphoJ**, pairwise comparisons are performed as part of CVA and Discriminant Function Analysis. In the **R**, pairwise comparisons can be performed using **lm()** or **adonis()** and, in the **shapes** package, using **resample** or **testmeans** (which is much faster).

Two-Group

Each group must be in a separate file, in standard (X1,Y1,…CS) format. The files are loaded separately by clicking **Load Data Set 1**, finding the file, then clicking **Load Data Set 2** and finding that file. As usual, you can display the data in various superimpositions, by clicking on your choice in the **Show Data** field below the visualization window. Be sure to select the correct baseline points before choosing plots or analyses that use a baseline superimposition (i.e. BC or SBR). After the files are loaded, all of the test options are active, so be sure that you have selected the right number of bootstraps before clicking one of these buttons.

To test the significance of the difference between samples using the partial Procrustes distances, choose **Goodall's *F* (Procrustes)**. For this test, the coordinates are superimposed using GLS with the specimens rescaled to unit centroid size (see Chapter 5). Again, values of *F*, the degrees of freedom, and P will appear in the results window. The distance between means is also reported, which is the Partial Procrustes distance. The program also allows you to test the significance of the difference between samples using Bookstein Coordinates – **Hotelling's T^2 (BC)**. The results window will report values for *F*, the degrees of freedom, and P. The results will also include the distance between means. This distance is the summed squared distances between corresponding landmarks, but this is *not* the minimized Procrustes distance because the landmark configurations are not in Procrustes (GLS) superimposition.

Next to the buttons for the analytic tests are buttons for two tests that use a bootstrap resampling procedure: ***F*-test, Procrustes**, which does a resampling-based version of Goodall's *F*-test and an ***F*-test, SBR,** which does a resampling-based *F*-test for coordinates superimposed by the Sliding Baseline registration. Before running the test, select the right number of iterations in the **No. of Bootstraps** box on the far left. The results, which will appear in the results window, will include the *F*-value computed for the original data set, then the **Significance level:..**, which is the fraction of iterations (in decimal format) in which *F* is greater than or equal to the value for the original data. If you use the ***F*-test, Procrustes** option, the output includes the (partial) Procrustes distance between means of the two data sets. If you selected SBR, the distance is not a Procrustes (minimized) distance because the specimens are not in Procrustes superimposition. In Chapter 11, we will return to the analysis of distances when comparing the lengths of ontogenetic trajectories.

In addition to the statistical tests, **TwoGroup** can plot the superimposed landmarks and the superimposed means (but only for the data sets 1 and 2). These plots can be modified using the **Symbols Control** pull-down menu, which allows you to change the red and blue symbols to black or gray, fill the symbols and increase their size. The plots of the differences between means can be edited using the options located on the **Difference Plot Options** pull-down menu. As usual, you can select from a variety of superimposition methods and types of displays, trim the grid and rotate the reference.

tpsRegr

The procedure for doing a pairwise comparison in **tpsRegr** is the same as described above for a multigroup analysis. The independent variable contains only one column because there are just two groups and there is one fewer column than groups. The first group is coded as **1**, the second as **-1**.

MorphoJ

To do a pairwise comparison in **MorphoJ**, input your data (coordinates and classifiers) and superimpose the shapes, as described previously for CVA. Select that data set and go to the **Comparison** menu. If you have only two groups to compare, go to the pairwise **Discriminant Function** menu. If you have several pairs of means to compare, go to the **Canonical Variates Analysis** menu. There is a difference between these two analyses because the **Canonical Variates Analysis** is based on all the groups, and it uses a pooled within-sample covariance matrix when maximizing the separation between groups relative to the within-sample covariance matrix. That pooled within-sample matrix is used when computing the Mahalanobis' distances and the statistics. In contrast, the **Discriminant Function** analysis is based only on the two groups being compared.

After selecting your procedure, you will then need to **select** your **dataset** and **data type** (**Procrustes coordinates**) and the classifier to be used for grouping. On the **Discriminant Function** menu, the box on the right allows you to **select specific pairs of groups** or you can check the box on the bottom left to **include all pairs of groups**. Beneath that is the box that you check if you want to do the permutation test (the test statistic is Hotelling's T^2). For CVA, the results include the **Mahalanobis** and **Procrustes** distances between means and the permutation test results. For Discriminant Function, the **Mahalanobis** and

Procrustes distances, parametric Hotelling's T^2, and permutation test results (if requested) are reported for each selected pair (as well as the classification results, discussed in Chapter 6).

PAST

In **PAST**, all the pairwise comparisons were done automatically when you did the MANOVA.

R

The shapes package includes two functions (**resampletest**, **testmeanshapes**) that do pairwise comparisons of means. **testmeanshapes** is far faster. Both allow you to use permutations or bootstrapping; the default is to resample with replacement (**replace = TRUE**) but you can change that to **replace = FALSE**. Both report Hotelling's T^2 statistic and Goodall's *F*-statistic, along with one that we did not discuss in the text (James T^2 statistic), and **resampletest** reports another test not discussed in the text (lambda-min) for 2D data. The results include the parametric *p*-values and resampling-based values. When the sample sizes are small relative to the number of landmarks, an adjustment is made to the within-sample covariance matrix. To run **testmeanshapes** on two samples, A and B, with 1000 permutations, write:

```
testmeanshapes(A,B, resamples = 1000, replace=FALSE)
```

You can try this with two of the datasets in the shape package, such as the male and female gorilla data (**gorm.dat**, **gorf,dat**). Or, if you did the MANOVA and now want to do pairwise comparisons between your groups, you can subdivide your data file by group, then put pairs together in the same file, put them in the shapes format, superimpose the pair of them and do the test on each pair.

For the male and female gorillas, combine the two data sets, which are already in the format of the shapes array, superimpose them and then divide them into the males and females.

```
gorilla<-array(c(gorf.dat,gorm.dat),dim=c(8,2,59))
gorillaProc<-procGPA(gorilla)$rotated
gorProcf<-gorillaProc[,,1:30]
gorProcm<-gorillaProc[,,31:59]
```

Then you can use **testmeanshapes()**, entering

```
testmeanshapes(gorProcf,gorProcm,resamples=1000,replace=FALSE)
```

You can follow this same procedure using your own data, selecting the specimens that belong to a group, combining them with the members of another group; then following the superimposition, dividing the combined groups into separate ones for purposes of doing the test. The third index of the array is position of each specimen in the file, so if your first group occupied the first 20 rows of your IMP file, their position in the array is:

```
myFirstGroup <-myData[,,1:20]
```

Finally, to produce the graphical output, we compute the mean shapes, then use any of the scripts for the graphics (see Chapter 5). For example, we could enter:

```
plot(gorProcf,gorProcm, type="n",asp=1, xlab="",ylab="",axes=F)
arrows(matr[,1],matr[,2],matt[,1],matt[,2],length=0.1,lwd=2)
points(matr, pch=21,col="black", bg="red", cex=1)
```

Literature Cited

Claude, J. (2008). *Morphometrics with R*. Springer, New York.

Oksanen, J., Blanchet, F. G., Kindt, R., Legendre, P., O'Hara, R. B., Simpson, G. L., Solymos, P., Stevens, M. H. H. & Wagner, H. (2011) vegan: Community Ecology Package. pp.

9

General Linear Models

We have good news and bad news about fitting complex statistical models to shape data. The good news is that it is, in fact, possible to fit and test even the most complex model without writing your own statistical program. The bad news is that the software you need heavily depends on the model. Two specialized morphometrics programs can analyze fluctuating asymmetry (FA), which is done by a two-way, mixed-model analysis of variance, with **Individual** and **Sides** as the two main factors, and **Individuals x Sides** as the interaction term (which is the term for FA). Both **MorphoJ** (Klingenberg, 2011) and **Sage** (Marquez, 2012) are designed for analyses of FA. Relatively simple models, such as a one-way multivariate analysis of covariance (MANCOVA) can be analyzed using **tpsRegr** (Rohlf, 2011). This program presumably can analyze any model, but the manual (and output) are rather terse and, as Rohlf says in the help file, the program is quite complex. **R** offers two approaches to analyzing complex multifactorial models, one being the multivariate parametric tests using **lm()**, and the other being distance-based permutational MANOVA, using **adonis()**, in the **vegan** package (Oksanen et al., 2011). **adonis()** permutes the raw data only (not residuals) and, although it does allow for permuting groups (**strata**), keeping them intact rather than permuting among the levels of the factor, the significance test always uses the residual (not variation among strata) as the denominator mean square. Recently, a new function enables testing two-factor nested models, **nested.npmanova()** in the **BiodiversityR** package (Kindt and Coe, 2005). But nested models with covariates and/or more than two factors present real difficulties and, in general, all mixed models (save the one for FA) require more specialized software.

That more specialized software is **DistLM** (Anderson, 2004) http://www.stat.auckland.ac.nz/~mja/Programs.htm. **DistLM** offers you exceptional control because you specify both the numerator and the denominator for every term in the model, and you can also use the raw data, the residuals of the reduced model or the residuals of the full model or the levels of another factor, in the permutation tests. **DistLM** produces what Anderson calls a “pseudo-*F*-ratio”, the *F*-ratio from a distance-based permutational MANOVA. The program, while very general, is also quite difficult to use; it can be especially nerve-wracking for poor or distractable typists because it runs in the DOS window. As a result, you cannot copy and paste control lines into the program and some typing errors will close the program. Based on my (MLZ) experience with multiple typos, mistyping a file name will not crash the program, but typing the wrong number when queried about the number of columns in a matrix will, as will replying **y** or **n** to a yes/no question instead of **1** or **2**. The program will also close if you try to load a file that is open in some other program or to write to a file that already exists. Perhaps more importantly, at least for good typists, is that **DistLM** requires you to load design matrices for each term, so you have to produce these as well as your shape data file.

We begin this chapter with the two programs that can analyze FA. We then discuss more general programs that can analyze any model that uses the residual mean square as the denominator in the *F*-ratio; these do not require you to produce a design matrix. We then discuss design matrices, which are required by both **tpsRegr** and **DistLM**. We conclude by explaining how to run these programs, focusing on **DistLM** because we understand this one better, both the details of the statistical approach and the mechanics of using the software. The final section of this chapter covers producing the graphical output.

Fluctuating Asymmetry (FA)

The two-way mixed model analysis of variance has been of special interest in evolutionary developmental biology and also in conservation biology because this is the preferred method for analyzing fluctuating asymmetry (FA). Two programs are specialized for the analysis of FA, one is the **Procrustes Anova** module in **MorphoJ** (Klingenberg, 2011) and the other is **Sage** (Marquez, 2012) in the Morphospice series (http://www-personal.umich.edu/~emarquez/morph/index.html). Both offer parametric and permutational tests of a generalized Goodall’s *F* (called a “**Procrustes Anova**” in both programs) plus a multivariate test that takes the covariance matrix into account (called a “**Procrustes Manova**” in both programs). In **MorphoJ**, it is also possible to include additional factors in the analysis so long as they are crossed. One important caveat: both programs will miscount the degrees of freedom when the data contain semilandmarks as well as landmarks. That is not a problem for the permutational tests, but the parametric *p*-values will need to be corrected.

MorphoJ (Procrustes Anova module, 2- and 3D)

In Chapter 3 (Superimposition), we discussed using **MorphoJ** to symmetrize your data. In this section, we focus on the statistical analysis of FA. To do the analysis, first load your data and classifiers (and covariate, if any). Then select that file in the **Project Tree**, go to the **Preliminaries** menu and do the **Procrustes superimposition**. Then go to the **Variation** menu and select **Procrustes Anova**. In the pop-up window, name your analysis, select the data set and classifiers for the effect (**Individual** and, in the case of matching symmetry, **Sides**). Then click on **Execute**.

On the **Results** tab you will see the tables for the Procrustes ANOVA, first for centroid size and then for shape, followed by the Procrustes MANOVAs for shape. **MorphoJ** offers parametric *p*-values for the Individual, Side and Individual * Sides (FA) terms. The results for centroid size come from the centroid sizes estimated by **MorphoJ** when it does the Procrustes superimposition; if the data were already superimposed in another program, centroid size cannot be estimated from the loaded data so the results for centroid size will not be correct. The degrees of freedom (and *p*-values) will need to be corrected if the data contain semilandmarks. Principal components of the Individual (symmetric) and Individual * Sides (FA) component can be obtained by selecting the desired component of the Procrustes ANOVA on the **Project Tree**, going to the **Variation** menu and selecting **Principal Components Analysis**. The graphical options are as described in Chapter 5.

Sage(2D)

In Chapter 3, we discussed using **Sage** to symmetrize data. In this chapter, we focus on the statistical analysis of FA. The first step is to load your data: **Sage** can read a tps file that contains either a scale factor or ruler, an IMP file (with centroid size) or a file of coordinates *x*1, *y*1, *x*2, *y*2, …*x*k, *y*k (without centroid size). **Sage** requires loading two protocols. One is the **Individual Protocol** that tells **Sage** which lines in the data file are for which individual. This file is a single column; the first individual in the file is coded **1**, the second is coded **2**, etc. If you digitized three photographs for each specimen (or for each side of each specimen), the column will contain three **1**s followed by three **2**s, etc. Presently, **Sage** can only analyze files that have the same number of replicates for every individual. If you are analyzing matching symmetry, the file will be organized differently because the right sides come first in the file and the left sides second, so the data for specimen 1 will not be contained in successive rows of the data file or in the **Individual Protocol**.

Analyses of object symmetry require a protocol file to tell **Sage** which landmarks on one side are paired with which ones on the other. This is the **Pair Protocol.** It contains two columns. The unpaired midline landmarks are listed first; the number for the landmark is in the first column and the number for its bilateral homolog (which does not exist for midline landmarks) is a zero **0**. After that come the paired landmarks, with one column containing the number for the left side, the other for the bilateral homolog on the right side.

Analyses of matching symmetry require a different protocol, one that identifies the side. The data should be sorted so that all the right sides come first in the file, followed by all the left sides. The **Sides Protocol** contains one column; filled with zeros (**0**) for all the right sides and **1**s for all the left. It does not actually matter whether the right or left comes first, so long as the side that does come first is **0** and the other is **1**.

To run **Sage**, **load** the data file; if you have a ruler in the file, specify its endpoints and scale. Then click **Proceed**. Then specify whether your data have object symmetry (like the **skull**) or matching symmetry (like **fly wings**). Click on the picture that represents your version of symmetry. Then **load** the two protocols; a pop-up window opens and says which protocol to load first, then second. Then select your analyses: **Procrustes ANOVA** and/or Procrustes **MANOVA,** and if you want **permutational-** as well as parametric *p*-values. Check the boxes for the desired analyses, and uncheck any defaults that you do not want. Enter the desired number of permutations. In the case of the **Procrustes MANOVA**, the parametric *p*-values cannot be obtained unless the sample size is larger than the degrees of freedom, and those degrees of freedom are calculated under the assumption that the data consist entirely of landmarks. If the sample size is too small, only the permutational *p*-values will be given.

Click **Run analyses**. The results will be shown in the blue box for the **Procrustes ANOVA** and in the yellow box for the **Procrustes MANOVA**. Use the pull-down menu to access the results for each term. The results can be saved to a text file using **Save results to text file**. To save the coordinates for the symmetric and FA components of shape, go to the **File** menu. First select your save options; the default for matching symmetry is to save the midline coordinates plus the coordinates for one half of the paired landmarks, i.e. those for one side. If that is your preference, you needn't do anything. Otherwise, select the option that you want (you can get the whole configuration as well as half if you want to do analyses with the half configurations but show the pictures on the whole). Go to the **Save fitted data** menu and select **Individual component (symmetry)**. The output file will be in IMP format. Then select **Interaction component (fluctuating asymmetry)**.

Multifactorial MANOVA/MANCOVA **(but not mixed models)**

adonis() R

adonis() is based on the methodological work of Anderson and colleagues (Anderson, 2001a, 2001b; Anderson and Robinson, 2001; Anderson & ter Braak, 2003; McArdle & Anderson, 2001). It therefore analyzes distance matrices, providing the "pseudo-*F* ratio" and the permutational *p*-value for each term in the model. The permutation procedure uses the raw data, or blocks of the data designated as exchangeable units (by the **strata** term in the argument of the function). As well as providing the pseudo-

F-ratios and permutational *p*-values, **adonis()** gives the R^2 for each term in the model, conditional on the terms preceding it (using Type I sums of squares).

To run **adonis()**, load **vegan** and your data. Make sure that all factors are assigned to factors, and covariates to vectors. Then enter the model. You also need to specify the distance metric; the default is Bray-Curtis so you need to stipulate that the distance is Euclidean. Our first five lines for an analysis of the effects of sex and region on shape read:

```
library(vegan)
data <- read.table(file.choose(),sep="", header=TRUE)
sex=as.factor(data[,1])
region=as.factor(data[,2])
shape=data[,-(1:2)]
```

When writing out the model, it is important to consider the order in which to list the terms because **adonis()** uses sequential (Type I) sums of square. To test the hypothesis that shape depends on size, on sex controlling for size, and on region controlling for size and sex: we would write the model as:

```
Result1= adonis(shape~ region+sex,method="euclidean")
```

To include the interaction term, we would write it as:

```
Result1= adonis(shape~region*sex *region,method="euclidean")
```

We can then see the statistical table by typing **Result1** at the command prompt (Table 1). And we can get the coefficients of the model, which are in **Result1$coefficients**. If we want the permuted *F*-values, they are in **Result1$f.perm**. To get the fitted values for the coordinates (to use for producing the graphics), also do the analysis using **lm()**, below.

Table 1. ANOVA table for a two-way distance permutational MANOVA produced by adonis()

	Df	**Sums of Squares**	**Mean Squares**	**F.Model**	**R^2**	**Pr(>F)**
Region	**1**	**0.005799**	**0.0057995**	**13.9097**	**0.10507**	**0.001**
Sex	**1**	**0.001604**	**0.0016040**	**3.8471**	**0.02906**	**0.001**
Region:sex	**1**	**0.000679**	**0.0006790**	**1.6285**	**0.01230**	**0.079**
Residuals	**113**	**0.047114**	**0.0004169**		**0.85357**	
Total	**116**	**0.055197**			**1.00000**	

Adding more terms to the model is straightforward. If you want only the main factors (not recommended), add the terms, separated by a +. If you want the interaction terms also, use *****.

lm(), manova()

The same models can be analyzed using the **lm()** and **manova()** function in the **stats** package (R_Development_Core _Team, 2011) in **R**. This gives a parametric *p*-value for a multivariate test. Unless your sample size is large relative to the number of variables, you will first need to reduce the dimensionality of the data, sometimes dramatically, by running a principal components analysis and using a subset of the PCs as the dependent variables. That introduces some arbitrariness into the analysis because the statistical results can depend (sometimes heavily) on the number of dimensions included in the analysis. An obvious question is: how many PCs to use? If your sample size is large, you could use as many PCs as there are degrees of freedom for shape, e.g. 2K – 4 for 2D landmark data. But if you have 15 landmarks and 85 semilandmarks and just 157 specimens, using 2K + L – 4 (=111) PCs will produce questionable results. If the PCA finds that 95% of the variation is explained by 25 PCs, we could use those 25 PCs, a reasonable number of variables to use when N = 157. To perform our analysis on the PC scores, we need to assign the principal component scores to a variable, **shape.PC**, and use that in our model. To do the analysis using **lm()** on the first 25 PCs, enter:

```
shape.PC<-prcomp(shape)$x[,1:25]
Result2<-summary(manova(lm(shape.PC~region*sex)))
```

Results2 contains the MANOVA table (Table 2). It also contains the fitted values (**Result2$fitted.values**) and the residuals (**Result2$residuals**). However, these fitted values and residuals are for the principal components, not our shape coordinates. To get the fitted values and residuals for the coordinates, we would write the model for the shape data, not the PCs.

Table 2. ANOVA table for a two-way parametric, multivariate MANOVA produced by summary(manova(lm(shape.PC~region*sex))); shape.PC comprises the first 25 principal components

	Df	Pillai	approxF	num Df	den Df	Pr(>F)
Region	**1**	**0.78261**	**12.8164**	**25**	**89**	**< 2e-16**
Sex	**1**	**0.34354**	**1.8631**	**25**	**89**	**0.01787**
Region:sex	**1**	**0.26116**	**1.2583**	**25**	**89**	**0.21498**
Residuals	**113**					

We can also do the statistical analysis with **manova().** To specify which test statistic you want, you include test = "testname" in the argument for the function.

```
Result3 <-manova(shape.PC~region*sex)
summary(Result.3, test = "Pillai")
summary(Result.3, test = "Wilks")
summary(Result.3, test = "Hotelling-Lawley")
summary(Result.3, test = "Roy")
```

nested.npmanova()

nested.npmanova() in the **BiodiversityR** package in **R** (Kindt and Coe, 2005) http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis is a function related to **adonis()** that is specialized for analyses of two-way nested designs. Unlike **adonis()**, **nested.npmanova()** tests the main factor against the nested term. The statistical significance of the *F*-ratio is tested by permuting entire blocks of the nested factor. Like **adonis()**, **nested.npmanova()** provides the pseudo-*F*, permutational *p*-value and R^2 values for the two terms.

To run **nested.npmanova()** we first need to load the **BiodiversityR** package and our data. We will again load two data sets, our shape data and our factors. But these are not the same two files that we use when running **adonis()**. First, the factors must be in a data frame with both row and column labels. And, because the header is necessary, a header for the shape data file is also necessary (else the program produces the error message that the two files have different numbers of rows). Another more consequential difference is that, based on our experience, the levels of the nested factors need to be labeled differently to be read correctly by the three programs in R that either run the analysis or output a design matrix. **adonis()** seems to read correctly the levels of the nested factor no matter how they are labeled. However, the program that outputs a design matrix (**model.matrix()**), requires that the nested terms be labeled sequentially within each level of the main factors. To read the design correctly, given 60 total litters, 20 nested within each of three diet classes, the codes would go from lit1 to lit20 for those in the first diet class, then again from lit1 to lit20 for those in the second diet class, etc. Coded that way, **adonis()** and **model.matrix()** correctly interpret the levels of the nested factor (and its degrees of freedom). But **nested.npmanova()** reads that as having only 20 degrees of freedom for the nested term. To be read correctly by **nested.pmanova()**, the nested factor must be labeled sequentially from 1 to the total number of levels, i.e. from lit1 to lit60. **adonis()** will also read that correctly, but **model.matrix()** will not.

The model formula for **nested.npmanova()** looks like a conventional model formula, such as those shown above for **adonis()**, **lm()** and **manova()**. But there is an important difference. The nested factor is *not* written Litter%in%Diet, which is the standard formula for a nested term in other statistical programs. Instead, it is indicated by its position within the formula; the main factor comes first, the nested comes second. The data frame containing the shape data (plus its header) is **PeroData**, the data frame containing the factors (plus headers and row names) is **PeroFactors**. This data file containing the factors must be included with the model formula.

```
library(BiodiversityR)
PeroData=read.table(file.choose(),sep="", head=TRUE)
PeroFactors=read.table(file.choose(),sep="",head=TRUE)
ResultPero=nested.npmanova(PeroData~Diet+Litter,data=PeroFactors,
method="euclidean", permutations=1000)
```

This function produces an ANOVA table (Table 3).

Table 3. ANOVA table for a nested permutational MANOVA, produced by nested.npmanova(). Nested anova for Litter nested within Diet

	Df	Sums of Squares	F	NPerm	Pr(>F)
Diet	**2**	**0.017913**	**5.7922**	**100**	**0.009901**
Litter	**55**	**0.085045**	**4.7210**	**100**	**0.009901**
Residual	**228**	**0.074677**			

Coding design matrices

The remaining two programs for GLM**, tpsRegr** and **DistLM**, require you to input a design matrix. Two programs will produce one for you, **model.matrix(),** a function in the **stats** package in **R**, and **XMatrix**, a DOS program by Anderson (2003). There are several methods for coding design matrices (also called X matrices). The two most commonly used are dummy coding and effect coding; both are used for unordered factors, meaning that they are not more or less of something but are truly categorical. For ordered factors there are two methods, Helmert (or reverse Helmert) and orthogonal polynomial coding. Helmert coding is used for ordered factors that do not have a meaningful distance from one level to the next; the first level may be “more” than the second but the factors are not incremental. For example, the levels of the factor might be wet, wetter and wettest, without signifying that the difference between wet and wetter equals the difference between wetter and wettest. Orthogonal polynomials are used when the ordered factors are equally spaced. For example, one level might be 100 degrees, the next 110, the next 120, etc.

The interpretation of coefficients of a model depends on how the factors are coded. That is the main reason for deciding to use one method rather than another. Dummy coding, which is also called treatment coding (as it is in **R**), treats one group as a baseline or standard to which all the others are contrasted. The mean of the baseline group is the intercept (constant). The coefficients for the factors are the differences between that baseline group and each of the others. For example, if the baseline group has a mean of 0.02 and the first treatment group has a mean of 0.01, the coefficient for that group would be -0.01. The coefficient for the second group is also its difference from that baseline group. If the baseline is a control group, each treatment is contrasted to the control group. Effect coding differs from dummy coding in that each group is contrasted to the grand mean. It is more useful than dummy coding when there is no logical baseline (i.e. control) group. It also has the advantage that the factors are orthogonal, which is not the case for dummy-coded designs, although the coding may need to be adjusted for unbalanced designs. This procedure is implemented by requesting **contrasts.sums** in **R** (because the sum of the codes for each column is zero).

Reverse Helmert coding contrasts the mean of each group to the mean of the groups that come before it. For example, the mean of the second group is contrasted to the mean of the first, and the mean of the third group is contrasted to the mean computed over the first two, and so forth. This is the procedure that is used in **R** if you ask for Helmert coding; usually Helmert coding is the reverse of that. Orthogonal polynomial coding contains information about the form of the curve along which the groups are equally spaced - the intercept is the constant (i.e. zeroth degree polynomial), the first coefficient is for the linear term, the second for the quadratic term, the third for the cubic, etc.

The terminology for coding can vary so it helps to know what the codes should look like. Dummy coding uses two values, either **1** or **0**. If there are four groups, there will be three columns of codes. In the first column, every individual belonging to the first group is coded as **1** and everyone else is be coded as **0**. In the second column, every individual belonging to the second group is coded as **1** and everyone else is coded as **0**. Similarly, in the third column, every individual belonging to the third group is coded as **1** and everyone else is coded as **0**. Usually, the constant in the MANOVA table is the mean of the fourth group, the one that has all 0s. If you are using dummy coding, put your control or baseline group last in the file.

Effect coding uses three values, **0**, **1** and **-1**. In the first column, the first group is coded **1** the last is coded **-1** and all others are coded **0**. In the second column, the second group is coded **1**, the last is coded **-1** and all others are coded **0**. In the third column, the third group is coded **1**, the last is coded **-1** and all the others are coded **0**. When the design is balanced, the codes for each factor sum to zero. The constant in the regression equation is equal to the grand mean of all the observations but, if the design is unbalanced, the grand mean is not equal to the mean of the means. The constant is the grand mean regardless of whether the design is balanced but when it is not balanced, the grand mean depends on the sample sizes for the groups. The regression coefficient for each term is the difference between the grand mean and the mean of the group coded **1**. The codes can be adjusted so that they sum to zero even when the design is unbalanced (see how dummy codes are adjusted for unbalanced designs, above).

The values for Helmert and reverse Helmert coding sum to zero, but the values depend on the number of groups in the design. When there are four groups, and the first is contrasted to the other three, members of the first group are coded as **0.75**, and members of the other three are coded as **-0.25**. Then, in the second column, the first group is coded as **0**, the second is coded as **0.66** and the third and fourth are coded as **-0.33**. In the third column, the first and second groups are coded as **0**, the third is coded **0.5** and the fourth as **-0.5**. Reverse Helmert coding reverses this scheme, e.g. in the first column, the first group is coded as **-0.5** and the second as **0.5** and the third and fourth are coded **0**.

In orthogonal polynomial coding, the degree of the polynomial depends on the number of groups in the design. If there are only two groups only the linear term can be fitted, if there are three groups, the linear and the quadratic can be fitted, and if there are four, the linear, quadratic and cubic can be fitted, etc. For a design containing four groups, in the first column, the first group would be coded **-3**, the second **-1**, the third **1** and the fourth **3**. In the second column, the first group would be coded **1**, the second **-1**, the third also **-1** and the fourth as **1**. These raw codes are then usually transformed to make the factors orthogonal and of unit length so this procedure is rarely done by hand.

Interaction terms are coded as the product of the main factor(s). So, if we want to code a two-way interaction between a continuously valued variable (e.g. size) and a categorical factor (e.g. sex), we would multiply the values for size by the codes for sex. This yields a single column because there is one column for size and one for sex. If there are two columns for one term and one for the other, the product has two columns. To code three-way (or higher-order) interaction terms, the coding is done in two or more steps. The first step is to multiply the first two terms, then that product is multiplied by the third term, etc.

Nested terms look clustered in a design matrix. The number of rows in the matrix equals the sample size, which equals the number of levels of the main factor ($A_1, A_2 \dots A_G$) times the number of levels of the nested factor ($B_1, B_2 \dots B_G$), times the number of individuals at each level of the nested factor. In a balanced design, the number of individuals is equal at each level of the nested factor and the number of levels of the nested factor is equal across all levels of the main factor. To make this more concrete (because the program you use might give you a very odd result, so you'll need to recognize that), we describe a hypothetical diet experiment in which one of three diets was fed to four litters of mice, each with a litter size of 2. So we have three treatments, four levels within each, two within each of those, for a total number of rows of 3 x 4 x 2 = 24. The total number of columns is the number of levels within each group minus one, times the number of levels in the main factor: (4-1) x 3 = 9.

The first three columns for the nested term are the litters nested within the first level of the main factor, diet (A_1). There are only eight individuals within that group so we can make a block of those eight rows, containing the individuals within that first level. We can similarly make a block of the eight rows within the fourth through sixth columns that contain the individuals within A_2. And we can complete this by making a block of the eight rows within the seventh through ninth columns that contain the individuals within A_3. Within that first block of eight for A_1, we code the two members of the first litter as **1** and the two within the last litter as **-1**. The other two litters in that block are coded as **0** and so is everyone else that is not in A_1. In the second column (still in A_1), we code the two siblings of the second litter as **1** and the two of the last litter as **-1** and everyone else nested within A_1 and everyone who is not within A_1 will be coded as **0**.

model.matrix()

To run **model.matrix()** you need your data frame and a model formula. You enter the right-hand side of the formula (i.e. to the right of the tilde) and the name of the data frame that contains the factors. If the factors were previously specified, such as by the line

```
Factor1 <- as.factor(data[,1])
```

then the information needed by **model.matrix()** is already available within the **R** environment.

```
xMatrix <-model.matrix(~ Factor1 * Factor2, data=Factors)
```

By default, dummy (treatment) coding will be used, so if you want an alternative you need to specify that, as in this example, which asks for effect coding.

```
xMatrix <- model.matrix(~ Factor1 * Factor2, data=Factors, contrasts =
list(Factor1="contr.sum", Factor2="contr.sum"))
```

It is possible to use different contrasts for different variables. For ordered factors, request either contr.helmert or contr.poly.

As mentioned above, if you are coding a nested design, repeat the values for the nested factor for each level of the main factor, e.g. if you are coding litters within diets, the first litter in the first group is lit1, and so is the first litter in the second diet treatment.

To check that the design matrix is correct, figure out how many rows and columns it should have (the number of columns should equal the degrees of freedom for the term(s) you have coded). To find out how many columns the matrix has, enter:

```
dim(xMatrix)
```

If there are far too many rows or columns, try again.

The first column is the intercept term, which you typically will not need for the two programs that input design matrices. In our experience, the matrix will not be properly ordered; the first column after the intercept term will be for the 10th nested level for each group, e.g. the 10th level for the nested term within the first level of the main factor, then the 10th level of the nested term within the second level of the main factor, then the 10th level of the nested term within the third level of the main factor. After that comes the 11th and then the 12th through 19th group, which is followed by the 2nd, then by the 20th. Put them in the correct order. You can delete the first column if you are using either **tpsRegr** or **DistLM** to analyze the data.

XMatrix

XMatrix (Anderson, 2003), like **DISTLM5**, runs in the DOS window. When you open the program, you will be asked a series of questions. The first of these is whether you want to produce: (1) X matrix codes for a single term (one of your main factors); (2) X matrix codes for an interaction term; (3) X matrix codes for a nested term; or (4) a single column of numbers for the levels of a factor (the category labels). To produce either the codes for the nested term or the single column of numbers for the levels of a factor, your design must be balanced. If your design is not balanced, make the list of column labels in a text-editor or spreadsheet. This list must be integers from 1 to the number of levels for the factor. Each term is saved to a file, so to produce the X matrix for the full model, you would open these separate files and paste the columns from each one into a spreadsheet. The advantage of producing the design matrix in **XMatrix** is that it offers an option for orthogonal coding.

When you want to produce an X matrix from the list of labels, you will be asked for the name of the file that contains them. Type that in (and do not forget to include the extension). You will also be asked for the output file name, so provide that (and do not name a file that already exists; the program will crash rather than overwrite an existing file). The program will then ask whether you want raw or orthogonal codes; if your design is unbalanced, you want orthogonal codes (if it is balanced, this does not matter). Then you are asked if you want to repeat that output matrix several times. If you answer 2 (for "no"), the output file is written and the program ends.

To code an interaction term, you need the X matrices for the factors. When asked, you enter the name of the file that contains the codes for the first term, and then you are asked how many columns that file contains. Then you are asked the name of the file containing the second X matrix, and you type in its name. You are then asked how many columns it has and the total number of rows in the matrix. You are then asked for the name of the output file. Then you are asked if you want raw or orthogonal codes.

Finally, you are asked whether you wish to repeat the output matrix. When you reply "2" (for "no") the codes are written to the output file and the program ends.

If producing the files for **DISTLM5**, you will need to create a file for each factor of interest. You will also need a file that contains the covariables, if any, which are the terms whose variation is not of interest. For example, if you want to know whether sex has an impact on shape, controlling for size, you would load the size X matrix as the covariables file. If you want to know whether elevation has an effect, controlling for both size and sex, then your covariable file would include an X matrix produced by pasting the column for size into the X matrix for sex terms. If there are any other terms in the model, you need an X matrix for them as well (otherwise the residual will be calculated from your term of interest and the covariables only). If you are using the mean square of another term as the denominator of the *F*-ratio, then you need the X matrix for that term as well. Finally, if you want to permute residuals from the full or reduced model, then you need an X matrix for the full or reduced model, which you would produce by pasting together the columns for all the terms in that model.

Running the models

tpsReg

Complex models can be analyzed in **tpsRegr** (Rohlf, 2011), although this is not as friendly as the other tps programs. To analyze a complex model, you load your data and a design matrix, then do one run that includes all the factors and interaction terms. By saving the residual sums of squares and doing another run using a reduced model, the difference in residual sums of squares allows for testing the statistical significance of terms left out of the reduced model. The same process is used when conducting a multivariate analysis of covariance (MANCOVA); this is the example of a complex model discussed most thoroughly in the help file. For this case, the first step is to test the significance of the interaction term, and then to test for the significance of a main factor, holding the covariate(s) constant. As described above, the first two runs test for the significance of the interaction term by comparing the fit of the full model (which includes them) to a reduced model that lacks the interaction term(s). Then (after clearing the residual sums of squares), the third run includes the covariate plus the factor and again the residual sums of squares are saved; in the fourth run, only the covariate is the independent variable and the difference in residual sums of squares are tested to determine whether the main factor is statistically significant. When sample sizes are large enough, **tpsRegr** provides both multivariate tests and a generalized Goodall's *F*-test as well as parametric and, if requested, permutational tests of the whole model (including all terms). The parametric tests presume that the data comprise only landmarks so the degrees of freedom for the *F*-ratio will be too large if your data include semilandmarks. Although laborious, the advantage of doing the analysis in **tpsRegr** is that you can produce the graphical as well as statistical results.

DISTLM5

To run **DISTLM5** (Anderson, 2004), you load your data and design matrices for each term in the model (so you will need to copy out the design matrices for each term and save each as a separate file). It is useful to put all your data and design matrices in the same folder as **DISTLM5** so that you can avoid typing in the full pathnames for all your files. You can load either the coordinates for your landmarks or the pairwise Procrustes distance matrix, such as the one output by **CoordGen7a**. There is a real advantage to loading a distance matrix, which is that you can avoid several queries that are devoted to converting the raw data matrix to a distance matrix.

You open the program by clicking on the executable. The first query is the name of the file containing your data. Type in the name of the file *including its extension* (*.txt). If you mistype it, the program will inform you that no such file exists and you can re-enter the name. The second query is the name of the output file. You enter the name of that file with its extension. Should you mistakenly give the name of a file that already exists the program will close rather than overwrite the file. The third query is whether the data file consists of raw data or a distance matrix. If your data file contains raw data, i.e. shape coordinates, you will be asked a series of questions about the organization of the file. The first is whether the rows are the samples and the columns are the variables or the rows are the variables and the columns are the samples. The next is the number of variables (columns) in the data and the next is the number of observations (rows) in the data. After that, you are asked for the desired transformation (if any). Then, after that, you are asked for your choice of standardization (if any). After that, you are asked for your desired distance metric; **Euclidean** is the third on that list. If you load a distance matrix, the only question you are asked is the number of rows. After successfully negotiating these several queries, you are finally asked whether you want to output the mean squares for the numerator and denominator under permutation.

Many of the questions prompt replies of “y” or “n” but the answers must be the numbers corresponding to “y” or “n”, which vary from screen to screen. If you type in the letter instead, the program will close.

After answering all the questions about your data, the first query about your model is whether it is an ANOVA or regression design. If you say it is an ANOVA design, you will then be asked whether you want to obtain a *p*-value using a Monte Carlo sample from the theoretical asymptotic distribution under permutation. After that, you are asked for the name of the file containing the design matrix (X matrix) for the factor of interest. Then you are asked for the number of columns in this matrix. After that, you are asked if there are any covariables in the model. "Covariables" is not strictly equivalent to "covariates" because covariables can be either continuous or categorical factors. In this context, covariables are the variables whose variation you want to take into account when testing the factor of interest. If you have covariables, you will be asked to name the file containing the design matrix for them, then asked the number of columns in that file. After that, you are asked if the denominator mean square (MS) for the test is provided by (1) the residual MS, with no other terms in the model; (2) the residual MS, but there are other terms in the model; or (3) the MS of another term in the model. If there are other terms in the model, or if you are using the MS of another term, you are asked for the name of the file containing the design matrix for those other terms and then the number of columns in the matrix.

The next series of questions concerns the permutation procedure. The first question is how many permutations you want (you can type in any number; the program is quite fast so you can reply **10000**). Then you are asked what general method of permutation you want; the possible answers are: (1) unrestricted permutation of the raw data or units; (2) permutation of residuals (full model); or (3) permutation of residuals (reduced model). If you select either the full model or reduced model, you are then asked for the name of the file containing the design matrix and then asked the number of columns in that matrix. After answering those questions, you enter an integer as the seed for the permutations (any value will do). Then you are asked whether you wish to permute units other than the individual observations. If yes, you are asked how many permutable units there are and then you are asked for the file containing the codes for those units, which is a single column, with integers specifying the unit that each individual belongs to.

When the program finishes the permutations, the results are shown on the screen and saved to the output file. The output file contains the information about the input files: the names of the data file, design matrix for the term of interest and design matrix for the covariables, the number of variables and the number of rows. The ANOVA table follows that, which includes the covariable degrees of freedom, sums of squares and mean square, the numerator degrees of freedom, sums of squares and mean square, and the denominator degrees of freedom, sums of squares and mean square. Below that, within the table, is the pseudo-*F*, the permutational *p*-value and the Monte Carlo *p*-value (if you requested that).

Below the table comes the information about the standardization and distance matrix, the permutation approach and number of permutations, and the integer chosen as the seed. The last line is the proportion of the variation explained by the factor of interest.

This sounds laborious, and it can be nerve-wracking if you are prone to typos or to forgetting such basic things as the number of columns in the design matrix for the reduced model. Nevertheless, of all the programs available for GLM, this one offers the highest level of control over both the model you fit and the permutation strategy, and the most complete detailing of results. In addition, the methods used by **DISTLM5** are thoroughly documented in Anderson and colleagues' publications.

Graphics

To depict your results, you will need to produce the shapes that show the contrasts between levels of the factors of interest, such as the means of the groups for one factor, after removing the variation explained by another. That requires doing a series of analyses, each producing the reference shape and the target shape for a factor (controlling for the other factors in the design). You can get these from the $fitted.values for each, one of the outputs of **lm()**. Converting them to the shapes format will then allow you to do the graphics for each factor.

The details of this procedure will obviously depend on your model. In Chapter 8, we showed how to obtain the shapes at the minimum and maximum values on the independent variable. And it is straightforward to get the mean shapes for each group. The only novel procedure is to remove the variation explained by a factor before computing the means for another one. One way to do this is to compute the grand mean, then fit the factor(s) whose variation you wish to remove from the data. The residuals from that analysis are then added to the grand mean. Then you can estimate the mean for each group, save the mean shape, convert it to the shapes format and use the scripts discussed in Chapter 5 to depict the shape differences.

In this example, we remove the variation explained by size and sex to show the variation explained by region, controlling for size and sex:

```
#compute grand mean
grandmean=t(apply(shapedata,2,mean))

#model with factors to remove
lm.remove<-lm(shapedata~size*sex, model=T,x=T,y=T,qr=T)
yhat.remove<-predict(lm.remove)
res.remove<-resid(lm.remove)

#output of residuals added to grand mean
sexOut<-res.remove
for (i in 1:nrow(res.remove)){
      out[i,]=res.remove[i,]+grandmean}
```

We then divide the data into the two regions, convert to the shapes format, calculate the means and draw the picture of the difference between means (you could, of course, compute the mean first and then convert the mean shapes to the shape format).

```
North=sexOut[region=="1",]
North=sexOut[region=="0",]
North=convertToShapes(North,2)
South=convertToShapes(South,2)

meanN=apply(North, c(1,2), mean)

meanS=apply(South, c(1,2), mean)
matt=meanN
matr=meanS
```

The effect is subtle, so we magnify it by a factor of five:

```
#magnify
matt1=matt+(matt-matr)*5

win.graph(8,6,12)
plot.new()
tiff(filename = file.choose(), width = 8, height = 6, units = "in", res =
1000)

plot(matr, type="n",asp=1, xlab="", ylab="",axes=F)
arrows(matr[,1],matr[,2],matt1[,1],matt1[,2],length=0.1,lwd=2)
points(matr, pch=20,col="black", cex=2)
title(main="Regional differences in chipmunk jaw shape")
title(line=-2,sub="magnified 5x",font.sub=3)
```

This produces the Figure W9.1.

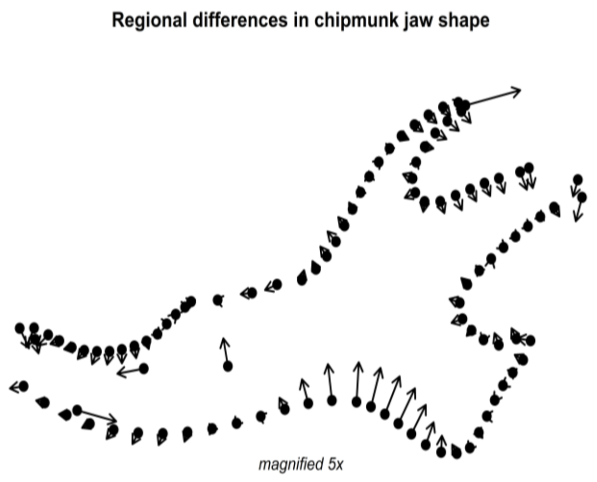


Figure W9.1. The impact of region, controlling for size and sexual dimorphism, on chipmunk jaw.

Literature Cited

Anderson, M. J. (2001a) A new method for non-parametric multivariate analysis of variance. *Australian Ecology* **26**: 32-46.

Anderson, M. J. (2001b). Permutation tests for univariate or multivariate analysis of variance and regression. *Canadian Journal of Fisheries and Aquatic Sciences* **58**: 626-639.

Anderson, M. J. (2003) XMATRIX: a FORTRAN computer program for calculating design matrices for terms in ANOVA designs in a linear model. pp. Department of Statistics, University of Auckland.

Anderson, M. J. (2004) DISTLM: Distance-based multivariate analysis for a linear model. pp. Department of Statistics, University of Auckland.

Anderson, M. J. & Robinson, J. (2001). Permutation tests for linear models. *Australian & New Zealand Journal of Statistics* **43**: 75-88.

Anderson, M. J. & ter Braak, C. J. F. (2003). Permutation tests for multi-factorial analysis of variance. *Journal of Statistical Computation and Simulation* **73**: 85-113.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies. pp. World Agroforestry Centre (ICRAF), Nairobi.

Klingenberg, C. P. (2011) MorphoJ: an integrated software package for geometric morphometrics. *Molecular Ecology Resources* **11**: 353-357.

Marquez, E. (2012) Sage. pp. University of Michigan, Ann Arbor.

McArdle, B. H. & Anderson, M. J. (2001). Fitting multivariate models to community data: A comment on distance-based redundancy analysis. *Ecology* **82**: 290-297.

Oksanen, J., Blanchet, F. G., Kindt, R., Legendre, P., O'Hara, R. B., Simpson, G. L., Solymos, P., Stevens, M. H. H. & Wagner, H. (2011) vegan: Community Ecology Package. pp.

R_Development_Core _Team (2011) R: A language and environment for statistical computing. pp. R Foundation for Statistical Computing,Vienna, Austria.

Rohlf, F. J. (2011) tpsRegr 1.38. pp. Department of Ecology and Evolution, State University of New York, Stony Brook.

10

Ecological and Evolutionary Morphology

Many of the questions about ecological correlates of shape variation or the functional implications of shape differences can be analyzed using methods described in previous chapters, particularly when the focus is variation within a species. However, if a question is about patterns of interspecific differences, these methods make assumptions about the independence of individuals and the structure of error that are violated by the hierarchical structure of phylogenetic relationships. In this chapter, we present methods that address this problem by incorporating information about phylogeny into the analysis of correlated changes.

In addition to correlations and trends, evolutionary studies may also be concerned with the magnitude of morphological diversity (aka disparity). For those studies, we present methods of quantifying disparity and testing whether samples are more or less dispersed than expected under selected random models. We also present methods of describing the hierarchical structure of disparity and evaluating how well the inferred structure represents distances between individuals in the data.

Independent Contrasts (MorphoJ)

The biggest difference between independent contrasts and other analyses you have performed (or seen performed) in **MorphoJ** is that you need to associate a phylogenetic tree with the data. That tree needs to be written in Nexus tree file format, which we summarize briefly below. The names on the taxa at the tips must match names written to a classifier variable, whether that classifier was submitted separately or extracted from identifiers. Once the tree is linked to the data, performing the analyses is rather simple – much more so than interpreting the results.

Nexus trees in MorphoJ

Nexus is a very flexible format for describing how to draw trees and is used in a great variety of phylogenetic software. In fact, it is so flexible most tree files output by phylogenetic analysis programs include a lot of stuff **MorphoJ** doesn't need. Fortunately, you can open that tree file in a text-editor and excise the few bits that actually describe the tree rather than the pretty colors and such used to draw the tree. A very simple example is written below; a more realistic example can be found in *jawstree.nex*:

```
#nexus
begin trees;
translate  1 A,
           2 B,
           3 C,
           4 D,
           5 E;
tree nuts = ((1:3,2:3):5,((3:2,4:2):3,5:5):3);
end;
```

The first line confirms that this is indeed a Nexus program. The next, **begin trees;** indicates that following commands constitute what is known as a tree block – a set of commands describing the tree. **translate…** indicates what names (A-E) will be used to draw the tree; these are also the names that are linked to the data that will be mapped onto the tree. These names must match your data set exactly in punctuation, capitalization, spelling and spacing. There is one exception: A_B will be read and stored as A B (yes, A<space>B), following Nexus conventions. Unfortunately, that means your classifier list should have A B, not A_B. The line beginning with **tree…** is the actual description of the tree, in Newick format. The name of the tree is **nuts**. Unlike some programs, **MorphoJ** will not accept a tree without a name. In Newick format, sister taxa are enclosed in parentheses and separated by a comma, the number after the colon indicates the branch length. In this example, (1:3,2:3):5 means that A and B (the "names" that go with 1 and 2) are sister taxa and both are distance 3 above their most recent common ancestor, and that hypothetical ancestor is 5 units above the next ancestral node (which happens to be the root in this example).

Note where the semicolons ";" are. These are the end-of-line codes. Do not forget them. The file won't run without them.

If you load a tree file with only these commands, **MorphoJ** will note in the Reports window that the tree file does not have a TAXA block. That is another way of reporting the names of the taxa. The translate command makes it unnecessary, and **MorphoJ** will happily go on without it.

Mapping Data onto the Tree

In the previous chapters, we described how to load your coordinates and covariates and link them. To compute contrasts, you need to load the tree file in the same way. On the **File** menu, go to **Import Phylogeny File**. After all the files are loaded and the superimposition has been computed, go to the **Comparison** menu and select **Map Onto Phylogeny**. Start with your covariate (probably centroid size) as this will produce something that looks familiar. Select weighting by branch length (which is the point, after all) and also use the tree as rooted. If you want the permutation test, choose a small number to start. The result produced in the plot window will show a tree on the vertical with the covariate values on the horizontal. If there is a lot of convergence, branches will cross each other to get where they are going. The node values are inferred by squared change parsimony optimization.

If you repeat this with the coordinates, your output graph will be a picture of a deformation between two nodes. Right click to choose different nodes. The shapes at nodes also are computed by squared change parsimony. The results tab will have diagnostics like the tree length and the permutation result if you selected it.

Analyzing contrasts

After the mapping has been performed, the project tree will show two new data sets: **BranchDiffs**… and **IndContrasts**… The first contains the differences between nodes you were just looking at; the second contains the independent contrasts you want to use to test hypotheses of correlated evolution such as evolutionary allometry. Select the independent contrasts set and then select regression from the **covariation menu**. The graphical results will be scores on the shape axis and size axis as before, but now these axes are contrasts. The deformation picture will again show the shape change that increases with size, but now this is the shape contrast that increases with larger size contrasts.

MorphoJ does not convert all the contrasts of the independent variable to positive values, so you will have to do that in another program. Export the regression scores and the contrasts for the independent variable; make sure to include the labels so you can correctly align the two columns. For any species in which the independent variable contrast has a negative value, multiply both that value and the matching regression score by –1. Combine the corrected values with the ones that did not need to be corrected and plot the results. It should now be clearer which contrasts have large values for both variables and which ones have large deviations away from the general trend. If you compute a regression for these data, bear in mind that the regression may explain a high proportion of the variation in scores on the shape axis, but this axis may not account for a high proportion of the variation in shape – it is only the axis through the shape space that is correlated with size.

You can also perform principal component analysis (PCA) on the contrast scores. Select the contrast data set as described above. Compute the covariance matrix for it and then compute the PCs using the covariance matrix of the contrasts.

Because PCA produces a rigid rotation of the original data, it is just as valid to compute contrasts on PC scores as on the original data. In fact, it may be easier and/or faster to compute contrasts of PCs if PCA produces an appreciable reduction of dimensionality. However, a major limitation of this approach is that **MorphoJ** will not draw deformations for contrasts of PCs. To get pictures of PCs of contrasts, you have to compute the contrasts first, then their PCs.

Phylogenetic comparisons in R

One limitation of independent contrasts is that the results are differences between nodes rather than residual values for the taxa at the tips of the tree. Revell (Revell, 2009) produces R-codes that generate those values for phylogenetic regression (as in analysis of evolutionary allometry) and phylogenetic PCA. As in PGLS, the initial steps of both programs take the branching pattern and branch lengths of a tree and compute the phylogenetic variance-covariance (VCV) matrix. Those codes are presented below, with some minor modifications.

One modification is to use the **read.nexus** command in **R**. This insures that the taxa in the various matrices are ordered as they are in the Nexus file **translate** command (not in the order they are encountered in the tree). Of course, this means your data files must be in the same order. In addition, for **read.nexus** to read correctly the list of names in the Nexus file, the first taxon must be on a new line. If the first taxon is on the same line as the translate command; its name will be ignored.

The regression program is the simpler of the two. It computes the estimated slope for each variable and the residuals from that slope for each individual.

```
# open ape and read in a tree in Nexus format
library(ape)
tree<-read.nexus(file.choose())
# compute the phylogenetic variance covariance matrix and invert it
C<-vcv.phylo(tree)
invC<-solve(C)
# read in your data, x for size (or other covariate), Y for shape data;
# convert each data set to a matrix;
# count the numbers of rows and columns in Y
x<-read.table(file.choose())
x<-as.matrix(x)
```

```
Y<-read.table(file.choose())
Y<-as.matrix(Y)
n<-nrow(Y)
m<-ncol(Y)
# create an empty matrix for the residuals from the regression
r<-matrix(,n,m)
# combine x with a column of 1s in a new matrix X
X<-matrix(,n,2)
X[,1]<-1
X[,2]<-x
# for each variable estimate the slope (beta) of its regression on x
# then compute the residuals from that regression for each individual
for (i in 1:m){
   beta<-solve(t(X)%*%invC%*%X)%*%(t(X)%*%invC%*%Y[,i]
   r[,i]<-Y[,i]-X%*%beta
   }
# done
```

The PCA program is a little more complex due to the need to compute an evolutionary VCV matrix from the tip data and the branch lengths. The eigenanalysis is performed on this matrix. (In addition to using **read.nexus** for the tree we have substituted **prcomp** for **eigen** and accordingly, replaced the eigenvalues computed by **eigen** with the squared standard deviations computed by **prcomp**.)

```
# open ape and read in a tree in Nexus format
library(ape)
tree<-read.nexus(file.choose())
# compute the phylogenetic variance covariance matrix and invert it
C<-vcv.phylo(tree)
invC<-solve(C)
# read in your shape data, and convert it to a matrix
# count the numbers of rows and columns in Y
Y<-read.table(file.choose())
Y<-as.matrix(Y)
n<-nrow(Y)
m<-ncol(Y)
# compute the ancestral states (root node)
# compute the evolutionary vcv matrix from that, the data
# and the phylogenetic vcv matrix
one<-matrix (1,n,1)
b <- solve(t(one)%*%invC%*%one)
c <- t(one)%*%invC%*%Y
a <- t(b%*%c)
delta <- Y-one%*%t(a)  # difference between tip and ancestor
V <- t(delta)%*%invC%*%(delta)*(n-1)^-1
# eigenanalysis of V, assign results to new variable names,
# make diagonal matrix of variances from standard deviations
out <- prcomp(V)
result <- NULL
result$Eval <- diag((out$sdev)^2)
```

```
result$Evec <- out$rotation
# compute scores in the species space
result$Scores<-delta%*%result$Evec
# and loadings, again using the evolutionary vcv matrix
ccv <- t(delta)%*%invC%*%result$Scores/(n-1)
result$Loadings <- matrix(,m,m)
for (i in 1:m)
   for (j in 1:m)
      result$Loadings[i,j] <- ccv[i,j]/sqrt(V[i,i]*result$Eval[j,j])
# done
```

The residuals from the regression and scores from the PCA can be plotted using routines demonstrated elsewhere. It is important to remember these results are scores in the original species space, not in a phylogenetically independent transformed space. Thus, they may be informative about differences among taxa and deviations from the dominant trends, but they are not phylogenetically independent values. It will still be necessary to perform subsequent analyses of these values using phylogenetic methods. However, performing the above routines first will insure that type I error rates are reduced to their nominal level when these residuals and scores are analyzed using phylogenetic methods (Revell, 2009).

Quantifying Disparity (DisparityBox – IMP)

The IMP program DisparityBox produces measures of disparity (as a variance) for groups, and comparison of disparities between groups. Analyses can be performed on a group that has no divisions, or on a group that has been divided into several subgroups. In either case, the analysis can be performed on the data as read, or size standardization can be performed on the data prior to analyzing disparities.

Single group analysis

Find the **Load Data** button in the middle of the right side of the screen, and then find your data. After your data are displayed in the small gray corner above the load button, perform Procrustes superimposition and compute the mean of the data set by clicking **Find Grand Consensus Mean (Specimens)**. Then, make the data set active by going to the **Active Set** box and clicking the **Up** button. Choose the number of bootstrap sets you want (probably just 100 for now), then go to the **1-Group Analysis** menu and click **Bootstrap Disparity within Group**. The disparity computed from the original data will be reported on the main screen, along with the confidence interval and standard error computed from the bootstrap sets. (Note that the choice of number of bootstraps is 100-500 not 100-2500.)

Multiple group analysis

If your data set includes several subgroups, each represented by multiple examples, you will first need to divide you data into several files. The example set *sqrlecol.txt* has mean jaw shapes of 31 squirrel species. These squirrels represent several clades as well as multiple ecological types. To examine whether evolution of flying squirrel or ground squirrels is associated with an increase of disparity in jaw shape, the sample was divided into three groups representing the basal tree squirrel clade (*sqrlcladeBT.txt*), the clade that includes flying squirrels (*sqrlcladeTF.txt*) and the clade that includes ground squirrels (*sqrlcladeGT.txt*). (There are more than 200 species of extant squirrels, so all three clades are grossly underrepresented in the demonstration data.) We can now evaluate the disparity of the combined data set as well as compare the contributions of the three subsets.

Clear the previous data and load each of the subsets. When done, click **List Loaded Sets**. The names of the files and the number of "specimens" in each will be written to the screen. This is handy if you have many data sets and have forgotten which ones you already loaded.

Next, click **Find Grand Consensus Mean (Groups)**. The program will compute the mean of each group, then the mean of the means. This insures that an overrepresented group does not unduly influence the estimate of the mean of the data set.

To analyze each subgroup, go to the **1-Group Analysis** menu and select **Distance Group mean to Grand mean**. In addition to computing the distance of that group's mean from the grand mean, the program will also compute the distances of each individual from the grand mean and from the group mean. These distances are used to compute both the mean of the distances and the square root of mean of the squared distances (RMS). The disparity of each group is also reported.

From the **Multi-Group Analysis** menu, select **Bootstrap Geometric Disparity** to get the disparity of the whole group, computed using the distances of the group means from the grand mean. **Bootstrap Geometric Disparity (MD, PD)** also reports the partial disparities of the subgroups and their standard errors. Partial disparities are based on the distances of individuals from the grand mean and, therefore, can be quite different from the disparities of those subgroup distances of individuals from the *group* mean. The standard errors are based on bootstrap resampling.

There are methods for testing whether subgroups differ in disparity. **Bootstrap Pairwise Difference in Disparity** generates bootstrap sets for each data set and computes the difference in disparity between the pair of sets generated each round. If the 95% confidence interval excludes zero, then the difference in disparity between groups is considered statistically significant. **Permute Pairwise Difference** permutes the two selected data sets and determines whether the difference in disparity of the permuted sets is greater than the difference in disparity between the original data set. The frequency of permutations in which this is true is the estimated probability that the disparities are different.

Disparity of size-standardized data

When taxa exhibit a substantial allometric trend, disparity in shape may be partly due to disparity in size or even differences in size sampling. To account for differences in size (or any other covariate), a regression can be performed and the residuals from the regression are used to compute the deviations from a standardized size. The analysis of disparity then can be based on the shapes in the size-standardized data.

To perform size disparity analyses on size-standardized data, **DisparityBox** needs the sizes (transformed to their natural logs) on which each data set will be standardized. These values must be written to a text file containing a single column of numbers, one value for each data set. So, for the analysis of three clades of squirrels, the size targets file will contain just three values, one for each line. The size values can be the same, or each can be different but, in the latter case, you need to remember the order of the numbers and input the coordinate files in the same order. (If you have a lot of groups standardized on different sizes, it may be easier to generate size-standardized data for each group in another program and perform the disparity analysis on the standardized data, using the buttons for unstandardized data.)

The list of size targets is loaded into **DisparityBox** after the data coordinates – click **Load Log Size Targets**. Then, when you click **List Loaded Sets**, the target sizes will be shown next to the sample size for each group. Click **Find Grand Consensus Mean**, as before. You then perform the analyses as you would for the unstandardized data, except now you select buttons with "Size Corrected" in the label.

Imaging Disparity

You probably noticed long before this point that **DisparityBox** provides little in the way of graphical displays. There is just a small plot of the superimposed shapes and sample mean, and you have already seen better versions of this plot generated by almost any other program. The lack of graphics is a reflection of the program’s function: to provide measures of the dispersion of a sample. Other programs provide descriptions and analyses of the dimensions in which the sample is dispersed; if the variation in your sample is concentrated in few dimensions, a PCA program may give you a suitable picture of the distribution of specimens in the space, and illustrations of shape changes in various directions within that space. Unfortunately, no program is going to give you a reasonable illustration of a distribution in four or more dimensions. And, even for dispersions in just a few dimensions, you should remember that the sum of two shape vectors often produces results that are difficult to visualize in advance.

Nearest Neighbor Analysis (DisparityBox – IMP)

In addition to quantifying the observed disparity, **DisparityBox** can test whether the observed dispersion is greater or less than would be expected under certain null models. The analysis can be performed for a single group (modeling the dispersion of individuals within the group) or for multiple groups (modeling the dispersion of group means).

Before requesting the analysis, you have to choose the parameters of the model, using the two buttons at the bottom of the screen. The upper button toggles between Gaussian and uniform models of a distribution. Gaussian uses the familiar Gaussian (bell-shaped) distribution – progressively fewer cases farther from the mean, with nearest neighbor distances increasing with distance from the mean. The alternative is a uniform distribution with the probability of occurrence the same everywhere in the space. The lower button toggles between methods of estimating the limits of the range from the observed sample. Sadler style range estimates are predicated on the notion the sample underestimates the true range by a factor predicted by sample size. The alternative is to use a multiple of the standard deviation – based on observation that, in normal distributions, the probability of specimens being found decreases rapidly with increasing distance from the mean.

Before starting, you also need to decide whether you want to perform the analysis on size-standardized data or not. **DisparityBox** can perform the size standardization for a multiple group analysis, so you just need to load a set of size targets, as for the analysis of disparity. However, there is no size standardization option for a single group Nearest Neighbor analysis, so if that is the analysis you want, you will need to generate the size-standardized data in another program and then load those data.

To run the analysis, load the data (and size targets if needed) and choose which reference shape to use (mean of mean or mean of individuals). Set the number of Monte Carlo simulations in the **Bootstrap Sets** window. Set the model parameters (Gaussian or uniform; Sadler or standard deviation- based range estimation). To analyze multiple groups, go to the **NN Model** menu and choose the first or second option, depending on whether you want the program to do size standardized or not. To analyze a single group, make the group active then go to the **NN Model** menu and select the last item (active group).

After you choose which analysis in the **NN Model** menu to run, the specified number of Monte Carlo simulations is run, using the specified model parameters. As with the analysis of disparity, results for the observed data and the summary results for the simulations are written to screen. The key results are the number of simulations with a smaller mean nearest neighbor distance (NND) and the pmean range. A large number of simulations with smaller mean NND than observed suggests over dispersion (observed NND greater than expected by chance), and a small number of simulations with smaller mean NND than observed suggests clustering or under dispersion (observed NND smaller than expected by chance). Similarly, if the average pmean is positive (observed NN tends to be farther away than the simulated NN) and the range excludes zero, the observed data set is significantly more dispersed than expected; if the average pmean is negative (observed NN tends to be closer than the simulated NN) and the range excludes zero, the observed data set is significantly less dispersed than expected.

Cluster Analysis

Clustering algorithms are included in **R** and in most major statistical packages. Running the algorithms is usually quite simple; the bigger challenge is deciding what the results mean.

After you have Procrustes superimposed data, you can perform a cluster analysis on the coordinates. If you have a large number of landmarks and semilandmarks, it might be more efficient first to perform principal components analysis and then do the cluster analysis on the PC scores. If you include all the PCs with non-zero eigenvalues, the results will be the same. In either case, format your data in a standard data-file format, like that used in IMP but without labels or any other variables (this means you remove centroid size, also). Select the type of clustering to perform; be sure to choose Euclidean distances for the dissimilarity matrix. Output will include a dendrogram, and may also include a text description of linkages. You may need to modify the plot to have tip labels at the same level (as in an ultrametric tree).

One option for testing whether the inferred clustering is meaningful is to use a k-means test. This test uses a criterion similar to that used in Ward's clustering to build the clusters and test whether they are statistically significant. One important difference between Ward's clustering and the k-means test is that the latter requires you to predict how many groups there are. It then builds clusters maximizing between group variance and minimizing within group variance. The results should indicate which individuals were assigned to each group and whether the groups are significantly different. You should understand that the k-means test could support the number of groups you inferred from the dendrogram produced by a cluster analysis, but those groups could have different members than you would have inferred from that dendrogram. Another option for testing the validity of groups generated by cluster analysis is to use a canonical variates analysis with cross-validation classification test. The MANOVA will test the hypothesis that there are different groups; the classification test will evaluate the assignments of taxa to those groups.

However you test the inferred groupings, it also is advisable to evaluate how well the cluster results accurately represent the hierarchical structure of the data. Commonly, this is done by performing the cophenetic correlation test, which compares the distances between specimens that were used to build the dendrogram and the distances between them on the dendrogram (their “heights” above the node joining them). The cluster analysis and the cophenetic correlation coefficient are easily generated in **R** from Procrustes superimposed coordinate data in IMP format (i.e. from your **CoordGen** output).

```
#Read your data into R by opening a window so you can navigate to the file and
select it.
mydata <-read.table (file.choose())
# count the number of columns and delete the last one (centroid size)
k<-ncol(mydata)
mydata<-mydata[,-k]
# compute the Euclidean distances between specimens
D1<-dist(mydata, "euclidean")
# perform the clustering
cluster<-hclust(D1, method="single")
# to use other clustering algorithms, substitute the name, e.g. "ward" or
"average" (for UPGMA)
# compute the distances along the branches
D2<-cophenetic(cluster)
# compute the correlation and write it to the screen
cor(D1,D2)
# draw the dendrogram
plot(cluster)
```

Literature Cited

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

11

Evolutionary Developmental Biology (1): The Evolution of Ontogeny

This chapter discusses the software for comparing ontogenies of shape and for integrating those studies with comparative studies of disparity. The software is still under development so some tests discussed in the textbook are not yet generally available. Some can be obtained by request from the author (and the ones that we requested are available here). Others are running in a beta version but they need further work before being released. We should soon have all the necessary software and we will update this chapter to keep pace with the software developments.

Comparative studies of ontogeny have taken one of two general approaches according to the aims of the study. The first aims to characterize how the ontogenies differ, without focusing on any one hypothesis; the second aims to test a specific hypothesis of interest, e.g. ontogenetic scaling. The two approaches differ in that the first progresses through a series of tests according to the results of the previous ones; whereas the second begins by testing the specific hypothesis of interest and may go on to consider alternatives when the hypothesis of interest is rejected. The two are clearly not mutually exclusive strategies but they differ notably in the sequence of tests. Rather than organizing this chapter according to hypotheses, we outline a sequence of tests that characterize how ontogenies differ.

The tests usually begin by conducting a multivariate analysis of covariance (MANCOVA). What happens next depends on the results of the MANCOVA. If the interaction term is *not* significant, the next step is to determine whether species follow parallel or overlapping trajectories. If they overlap, the question is whether the two exhibit ontogenetic scaling or dissociation between size and shape. If the interaction term *is* significant, the next step is to determine whether species differ in the direction of ontogeny or only in its length. If the direction does not differ, the next step is the same as that taken when the interaction term is not significant - to determine whether species follow parallel or overlapping trajectories and, if overlapping, whether they exhibit ontogenetic scaling or dissociation between size and shape. After identifying the differences in ontogenies, it becomes possible to examine how those differences affect disparity. One question is whether disparity increases or decreases over ontogeny and another is which of the changes in ontogeny elevate or depress disparity.

We begin this chapter by reviewing the first step: conducting the MANCOVA introduced in Chapter 9 (General Linear Model). We then organize the remainder of the chapter around the tests that are done if the interaction term is significant and then discuss the tests that are done if the interaction test is not significant. We then briefly discuss the software available for studies of disparity but much of this remains under construction. We conclude with a summary of the software available for standardizing data, which can be used to produce discrete size classes for use by the programs that compare directions and lengths of vectors ending between means.

The first step: MANCOVA (R)

A permutational MANCOVA can be done in **R** using **adonis()**. To run the analysis, first load your data, in standard format (*x*1, *y*1, *x*2, *y*2…*xk*,*yk*); for 3D data, the *z*-coordinate would follow the *y*- for each landmark. You will need two additional columns, one a code for the species, the other a column of centroid sizes. For a file that contains the species factor in the first column, centroid size in the second and shape coordinates in all other columns, we would begin with the following lines:

```
data <- read.table(file.choose())
species <- as.factor(data[,1])
size <-as.vector(data[,2])
shape <-as.matrix(data[,-(1:2)])

#you will want to log transform size
LCS=log(size)

require(vegan)
result1<-adonis(shape ~ LCS*species,method="Euclidean")
result1
```

If you run this script with the piranha ontogenetic series (pirOnt.txt), you will get Table 1. The interaction term clearly is statistically significant, so we now compare: (1) directions of ontogenetic change; (2) lengths of ontogenetic trajectories; and (3) shape at the outset of juvenile growth.

Table 1. MANCOVA in adonis(); ontogenetic series of nine piranhas (file = pirOnt.txt)

	Df	Sums Of Squares	Mean Squares	F.Model	R2	Pr(>F)
LCS	**1**	**0.23523**	**0.235233**	**192.234**	**0.12176**	**0.001 *****
Species	**8**	**1.10924**	**0.138655**	**113.310**	**0.57414**	**0.001 *****
LCS:species	**8**	**0.13230**	**0.016538**	**13.515**	**0.06848**	**0.001 *****
Residuals	**372**	**0.45521**	**0.001224**	**0.23562**		
Total	**389**	**1.93198**		**1.00000**		

When the interaction term *is* significant: Comparing the directions of ontogenetic change

Several programs can test the null hypothesis that the direction of ontogenetic change does not differ between species. The programs, however, are not interchangeable. They differ in at least two important respects. One is that some programs are suited to data from an ontogenetic series, in which the independent variable (usually size) is continuous. The others are suited to data from discrete classes, such as the youngest and oldest age classes. The second difference is the permutation strategy.

VecCompare7 (IMP) compares the direction of ontogenetic change estimated by regressing shape on size, and tests the null hypothesis that the angle between two ontogenetic trajectories does not exceed the angle that could be obtained by resampling a single species. The test works by producing two bootstrap sets for each species at each iteration of the resampling procedure. For each pair of bootstrap sets from a single species, the regression model is fitted to the species' data, the residuals from that model are randomly drawn twice, producing the two bootstrap sets of residuals per species. The residuals are added to the expected values for the observed sizes and the regression model is refit to each bootstrap set. The angle is computed between the regression vectors for the two bootstrap sets, and the process is iterated, generating the null distribution of angles. If the observed angle exceeds 95% of the angles obtained by resampling, the null hypothesis is rejected. The comparison is done to the species that produces the larger range of angles. **VecCompare7** also does a test of the difference between two angles, which can determine whether the difference between one pair of species exceeds that between another pair of species.

Another program, by Piras and colleagues (2011), **Common.slope.test()** runs in **R**. Piras kindly provided the unpublished script for this and several other functions discussed in this chapter, which are part of a planned package for ontogenetic studies in **R**. Like **VecCompare**, **Common.slope.test()** works with the coefficients of a regression equation. For each species, shape is regressed on centroid or log centroid size, and the coefficients of the two regression equations are compared between two species. That comparison is done by measuring the distance between two vectors, a measure highly correlated with the angle between the vectors. The test of the statistical significance in that distance is done by permutation of residuals from the regression but, unlike the tests discussed below, the permutations are restricted to the two species being compared. The observed value is obtained by fitting regression models to two species and computing the distance between the vectors of coefficients of the model; the null distribution is obtained by randomly permuting the group codes and fitting the models to the randomized data sets. The distance between the vectors of coefficients is computed between the randomized data sets, at each iteration, producing a distribution of the distances under the null. The *p*-value is then determined by the proportion of values in the null distribution that are either as large as, or larger than, the distance between the observed data sets.

Two related programs for comparing phenotypic trajectories by Collyer and Adams (Collyer & Adams, 2007) and Adams and Collyer (Adams & Collyer, 2009) compare trajectories between means, such as the means for the youngest and oldest age classes. These programs test the null hypothesis that the angle between species is no greater than would be expected by chance, using a randomization of the residuals from a reduced model (the model that excludes the interaction term) to generate the null distribution. Under the null, the residuals from the reduced model are exchangeable units. They are permuted, or rather, the group codes are. The residuals, randomly assigned to species, are added to the expected values for the mean. The full model is then used to calculate the predicted values from the random data. Repeating that procedure numerous times yields the null distribution under the assumption of the null hypothesis, which is that the reduced model is true. The *p*-value is then determined by the proportion of values in the null distribution that are either as extreme or more extreme than the observed value (which derived from the full model). The two scripts differ because the one by Collyer and Adams compares trajectories that have only two points (e.g. youngest and oldest) whereas that by Adams and Collyer compares trajectories measured at two or more points, such as ontogenies sampled at multiple ages. Even if you have multiple age-classes, it is worth looking at the code for the two-state trajectories because it is so meticulously documented. If your data come from an ontogenetic series, but you want to use this program, you can use one of the standardization procedures we provide at the end of this chapter.

We note that there is also a comparison of vectors available in **MorphoJ**, which differs from the other four programs not only because it uses a parametric test but also because it tests the null hypothesis that the vectors are no more *similar* than expected by chance. The observed angle is compared to those obtained by drawing random vectors from a uniform distribution on a hypersphere, a test based on a recent paper by Li (2011).

Running VecCompare7 (IMP)

Unlike other programs in the IMP series, **VecCompare** does not do a Procrustes superimposition when you load the data. So, if you have not already done so, do the superimposition on both species (including them in a single file) and save that file in IMP format, with centroid size in the last column. Then separate them into two files and load the first one, clicking on **Load File1** and opening the first file, then do the same for the second file using the **Load File 2** button. Then go to the **Regression function** window (below the **Exit** button) and decide whether you wish to do the regression on log centroid size (**LCS**) or centroid size (**CS**); the default is log centroid size. Select the desired number of bootstraps in the **Number of Bootstrap Sets Used** box below (the default is 100). Then select **Compute Growth Vector**. When the test is done, the results appear in **Results** window. These can be saved to a file by clicking on the **Save Results** button beneath the **Compute Growth Vector.** The saved results include the two vectors, the angle between them and the 80%, 90% and 95% confidence intervals (the upper limits on the range) for the angles obtained by resampling within each of the species. If you save the results to a file (rather than copying them from the Results window and pasting them into a file), remember that saving results from two analyses to the same file will overwrite the first; the files are not appended.

To do the test for a difference between angles, first load the two species whose angle will be one of the angles compared; if you have not already done so, load the first one, clicking on **Load File1** and opening the first file, then do the same for the second file using the **Load File 2** button. Then go to the **File** menu on the upper left of the toolbar, where you will see the options to load **File 3** and **File 4**. One of these two files could be the same as **File** 1 (or 2). Then go to the **More Statistics** menu on the toolbar and select **Bootstrap Test of a Difference in Angles**. In the **Results** window, you will see the results for each pair of species, including the angle between them and the 95% confidence interval on that angle. Below, you will see the 95% confidence interval on the difference between angles. If that confidence interval includes zero, you cannot reject the null hypothesis that the two angles are equal. If both the lower and upper bounds of the confidence interval are positive numbers, that means that the first angle is significantly larger than the second one; conversely, if the upper and lower bounds are both negative, it means that the second angle is significantly larger than the first (the second is subtracted from the first). These results cannot be saved to a file so copy them to the clipboard and paste them into a file.

Testing for a difference between two angles: **VecCompare7** also includes a test for the difference between two angles, testing the null hypothesis that the two angles are equal. This is done by resampling within each species, computing the angle between each pair of species, then computing the angle for each, then computing the difference between two angles.

Running common.slope.test() (R)

To run **common.slope.test()** you can use pirOnt because the program needs the same information used by the MANCOVA. The file should contain a code for species, values for centroid size (or log centroid size) and the shape data for the ontogenetic series. We run this analysis using pirOnt. The argument for the function is the name of the object containing the size data, the shape data and the grouping variable (species), in that order. To run the analysis, select the whole script, read the function into **R**, then enter:

```
Results2 <-common.slope.test(LCS,shape,species,nperm=1000)
Results2
```

The results for the piranhas are shown in Table 2.

Table 2. Results for common.slope.test() (file = pirOnt). Angles and *p*-values are rounded to three decimals to enhance readability
$p.value

	1.000	2.000	3.000	4.000	5.000	6.000	7.000	8.000	9.000
1.000	0.000	0.331	0.025	0.054	0.138	0.034	0.001	0.004	0.044
2.000	0.000	0.000	0.010	0.004	0.797	0.567	0.143	0.318	0.375
3.000	0.000	0.000	0.000	0.029	0.015	0.001	0.001	0.001	0.075
4.000	0.000	0.000	0.000	0.000	0.038	0.005	0.001	0.001	0.232
5.000	0.000	0.000	0.000	0.000	0.000	0.220	0.003	0.027	0.082
6.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.016	0.058
7.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.103	0.003
8.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002
9.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 2. Continued

$obs.diff

	1.000	**2.000**	**3.000**	**4.000**	**5.000**	**6.000**	**7.000**	**8.000**	**9.000**
1.000	**0.000**	**0.027**	**0.043**	**0.048**	**0.026**	**0.033**	**0.047**	**0.039**	**0.036**
2.000	**0.000**	**0.000**	**0.050**	**0.049**	**0.017**	**0.019**	**0.033**	**0.027**	**0.032**
3.000	**0.000**	**0.000**	**0.000**	**0.042**	**0.048**	**0.056**	**0.068**	**0.065**	**0.040**
4.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.047**	**0.055**	**0.068**	**0.065**	**0.035**
5.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.021**	**0.036**	**0.031**	**0.029**
6.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.023**	**0.021**	**0.034**
7.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.014**	**0.048**
8.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.046**
9.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**

Running Phenotypic Trajectory Analysis (R)

Both programs for phenotypic trajectory analysis can be run using the same data files. However, herein, we concentrate on the multistate trajectory analysis because it is the more generally useful program. For an analysis of ontogenetic trajectories, the input shape data consist of multiple species (or other sampling units) measured at two or more developmental stages. If you measured ontogeny continuously rather than at discrete stages, you can still use these scripts but you will first need to produce the discrete stages (more on this below). The data should include a column of codes for the species' identity and a second column of codes for the level of the factor. For a two-state trajectory, there will be only two levels of the factor ("1" and "2"); for multistate trajectories there will be as many as the number of samples obtained over the ontogeny. The shape data should be formatted with each row corresponding to an individual and each column to a coordinate (IMP format without centroid size, or **MorphoJ** format without the identifier). If the two factors and the shape data are in the same file, the factors should be removed from the shape data. For the piranha data file that contains shape data for two states ("Piranhas"), the species codes are in the first column, and age-class coded in the second. The initial lines are:

```
data<-Piranhas
taxa<-as.factor(data[,1])
level<-as.factor(data[,2])
Y<-as.matrix(data[,-(1:2)]) ## Removes factor columns from rest of data
taxaBylevel<-as.factor(paste(taxa,level))  #for LS means at each level
Y<-prcomp(Y)$x #TO USE IF > 2 variables: for plotting PC axes at the end
```

"Taxon" and "level" are thus both factors and **Y** is the matrix of shape coordinates. To run the program, you first read in your data (or "Piranhas"). There is, however, one caveat - if you measured many semilandmarks, far too many for the number of individuals in your file, you should rewrite the line that replaces **Y** by the principal components of **Y** to reduce the dimensionality of your data; if *k* is the number of PCs that you wish to include in the analysis, rewrite this as:

```
Y<-prcomp(Y)$x[,1:k]
```

Then select the code, and run it. The results for the analysis of piranhas are in Table 3.

Table 3. Results for phenotypic trajectory analysis (R)(file = Piranhas). Angles and *p*-values are rounded to three decimals to enhance readability
> trajdir.obs

	[1,]	[2,]	[3,]	[4,]	[5,]	[6,]	[7,]	[8,]	[9,]
[1,]	0.000	38.590	32.985	46.036	37.151	48.678	70.255	58.989	44.893
[2,]	38.590	0.000	38.388	45.733	26.309	33.064	54.569	48.194	39.111
[3,]	32.985	38.388	0.000	34.798	38.065	47.780	67.979	64.886	31.965
[4,]	46.036	45.733	34.798	0.000	44.968	55.594	76.314	73.411	30.801
[5,]	37.151	26.309	38.065	44.968	0.000	34.097	56.798	51.443	35.222
[6,]	48.678	33.064	47.780	55.594	34.097	0.000	39.051	39.971	41.829
[7,]	70.255	54.569	67.979	76.314	56.798	39.051	0.000	22.637	65.671
[8,]	58.989	48.194	64.886	73.411	51.443	39.971	22.637	0.000	63.736
[9,]	44.893	39.111	31.965	30.801	35.222	41.829	65.671	63.736	0.000

> POrient

	[1,]	[2,]	[3,]	[4,]	[5,]	[6,]	[7,]	[8,]	[9,]
[1,]	1	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
[2,]	0.001	1	0.001	0.001	0.001	0.001	0.001	0.001	0.001
[3,]	0.001	0.001	1	0.001	0.001	0.001	0.001	0.001	0.001
[4,]	0.001	0.001	0.001	1	0.001	0.001	0.001	0.001	0.001
[5,]	0.001	0.001	0.001	0.001	1	0.001	0.001	0.001	0.001
[6,]	0.001	0.001	0.001	0.001	0.001	1	0.001	0.001	0.001
[7,]	0.001	0.001	0.001	0.001	0.001	0.001	1	0.001	0.001
[8,]	0.001	0.001	0.001	0.001	0.001	0.001	0.001	1	0.001
[9,]	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	1

Running Compare Vector directions (MorphoJ)

To do the comparison in **MorphoJ**, load your shape data, with the specimen IDs in the first column, load the classifier file, also with specimen IDs in the first column and the classifier variable in the second (use headers in this file, naming the variables (e.g. ID, Species)) and load the covariate file with the classifiers in the first column and the values for log centroid size in the second (also use headers in this file, naming the columns, e.g. ID, LCS). Then, on the **Preliminaries** menu, select **New Procrustes Fit**. Then return to the **Preliminaries** menu and select **Subdivide Dataset By**, subdividing by the classifier to produce the data set for each species. Then go to the **Covariation** menu to do the regressions for all the species; selecting first the data set for the dependent variable (the species you want to analyze), the data matrix (Procrustes coordinates) and the variables (Procrustes coordinates). Do the same for the independent, selecting the data set for the same species, the data matrix (Covariates) and the variable (LCS). Remember to name the analysis so that you can find the regression results for each species on the next menu. Execute the analysis. After completing the regressions for all the species that you want to compare, go to the **Comparison menu** and select **Compare Vector Directions**. Select the first of the two vectors that you want to compare. These will all be labeled according to the names you gave the regression analyses, e.g. denticulate regression, altuvei regression. Then select the second vector and **Execute**. The result will consist of the angle between the two vectors and the *p*-value for the null hypothesis of no similarity between them.

When the interaction term *is* significant: Comparing lengths of ontogenetic trajectories

Two programs in the IMP series can compare lengths of ontogenetic trajectories to determine if they differ: **DisparityBox7** and **Two-Group7**. In addition, the two programs for **Phenotypic Trajectory Analysis (R)** discussed above compare lengths as well as directions, using permutations of the residuals to obtain the null distribution. Of these four programs, only **DisparityBox7** uses a regression model to measure lengths. The other three use the difference between means. **Disparitybox7** calculates the length of ontogenetic vectors by measuring the distance between juveniles and adults of the same species at two selected sizes, producing the expected values at those sizes from a regression equation. The calculations are based on a procedure similar to that used in **VecCompare**, i.e. fitting a regression model, calculating the residuals, drawing them at random and adding them to the expected values, then refitting the model, adding the residuals to the expected values for shape at the requested sizes. This procedure is done at each iteration of the bootstrap before calculating the distance between the two shapes, giving a confidence interval for the length of the trajectory that incorporates the uncertainty of the regression. **DisparityBox7** always regresses on the log centroid size. **TwoGroup7** and the **Phenotypic Trajectory Analysis** programs use the regression equation to produce the expected values at given sizes, to which the residuals are added, but this has to be done prior to running the programs and the uncertainty of the regression is therefore not taken into account.

Running DisparityBox7 (IMP)

To estimate the length of a single trajectory, first decide what sizes you want to use as the juvenile and adult sizes. These should be in units of the natural log of centroid size. Prepare the **Log Size Target List** by listing the desired sizes in a single column. A list for this comparison would look like this:

```
3.232
5.622
```

Save that as a text file. Then load the same data file twice using the **Load Dataset** button. Then go to the **File** menu and select **Load Log Size Target List** and select the size target list that you just prepared. You now need to compute the mean. **Select Find Grand Consensus Mean(Groups),** although in this case it does not matter at all which of the two options you use - the distinction only matters if sample sizes differ across different groups and you want the mean of the means rather than the grand mean. Then go to the **Multi-Group Analysis** menu (you are measuring the distance between groups, not the variation within a single one). **Select Bootstrap Size-Corrected Disparity**. Because this program is designed for analyses of disparity, which is measured as a variance (see Chapter 10), the result will be a squared distance, not a distance, so take the square root of the distance and of the distances at the upper and lower bounds of the confidence intervals.

When loading many files, especially if loading them twice (such as when measuring disparity at two developmental stages), it is easiest to make a chain file (batch file), listing the files in the order that you want them loaded (the names must include the extensions) and, if the chain file is in a different folder than the data files, the pathnames for the data file should be included in their names). That list should be in the same order as the list of target sizes because **DisparityBox** will compute the mean shape for the first species in the chain file at the first target size in the size target list. When preparing the size target list for different species, use the sizes of those species at corresponding developmental stages. For example, if comparing the disparity of juveniles of mice and rats to the disparity of adult mice and rats, use sizes for mice and rats at comparable developmental stages. These will be different sizes for the species because an infant rat is approximately the size of an adult mouse.

After preparing these two files, go to the **File** menu on the toolbar and select the last option, **Load From Chain File List**. Select the file that contains your list of files to load. When those files are loaded, go back to the **File** menu and select **Load Log Size Target List** and load your list of log centroid sizes. You can, however, load each data file one by one, using the **Load Data set** button; just make sure that you load them in the same order as their desired sizes appear in the size target list. Then select **Grand Consensus Mean (Groups)** unless you prefer to use the grand mean. Then select **Bootstrap Size-Corrected Disparity**. The results show up in **Results**. The results can be saved to a file; go to the **File** menu and select the first option. To save the results from the second analysis to the same file, select the second option, **Append Results to File**. You will be warned that the file exists and will be overwritten; ignore that, appending the file does not overwrite the first file.

Unfortunately, **DisparityBox** does not bootstrap the difference in disparities. **Two-Group** does, but it works with means rather than regression equations.

Running Two-Group7 (IMP)

To compare the distance between length trajectories you need four data files. If you are comparing the length of the trajectories across species, one file should contain the juveniles of that species, the second the adults of that species, the third the juveniles of the second species and the fourth the adults of the second species. You could also compare the lengths of the trajectory between infants and weanlings of one species to that between weanlings and adults of the same species. In that case, you would need only three files because you would load the file of weanlings twice. Load the first data set using the **Load Dataset 1** button and load the second data set using the **Load Dataset 2** button. Then go to the **File** pull-down menu and use **Load Group 3** and **Load Group 4** to load the second pair of samples. Now go to the **More Stats** pull-down menu and select **Bootstrap Distances 1+2 vs 3** + **4**. When the iterations are completed, the results will show up in the **Results** window. These results will include the Procrustes distance between means 1 and 2, the 95% range and standard error for the distance. Scrolling down you will see the same information for the distance between means 3 and 4. After that you will come to the 95% range for the difference between the distances. If this includes zero, then the distances are not significantly different. If both ends of the range are positive numbers, the first distance is significantly longer than the second; conversely, if both ends of the range are negative numbers, the second distance is larger than the first.

Phenotypic Trajectory Analysis (R)

The lengths of trajectories are calculated and tested along with the differences in direction. The test for a difference in length need not be requested specifically. Running the analysis to compare lengths is the same as running the analysis to compare directions, already covered above.

When the interaction term is *not* significant (or trajectories do not differ in direction): Comparing elevations

Several programs address the related question of whether ontogenetic trajectories differ in their elevation or instead overlap and, if they overlap, whether they overlap in size-shape space (as expected under the hypothesis of ontogenetic scaling) or instead overlap only in shape space (as expected under the hypothesis of size-shape dissociation). One test, by Piras and colleagues tests for a difference in the intercept of the allometric equations. The other tests, used by Gerber and Hopkins (2011), implement tests proposed by Mitteroecker and colleagues. (2005). Gerber kindly provided the unpublished scripts. These tests are intended to distinguish between ontogenetic scaling and heterochrony. Ontogenetic scaling is formulated in terms of two trajectories that overlap in both shape space and in size-and-shape space whereas heterochrony is formulated as two trajectories that overlap only in shape space. The hypothesis of ontogenetic scaling is tested by randomly permuting the summed squared residuals from the regression (i.e. by randomly permuting the group codes). The number of cases in which the value obtained by permutations is less than the observed one is counted. The *p*-value is determined by the proportion of values in the null distribution that are either as small as or smaller than the sum of squared residuals in the observed data. If the observed value is large, the null hypothesis (of ontogenetic scaling) is rejected. Heterochrony is tested by random permutations of the sum of squared distances between each individual and *its nearest point along the regression line*. These sums of squared distances are permuted and, again, if the observed value is large relative to the random values, the null hypothesis (of heterochrony) is rejected.

int.test() (R)

The test for the same intercept, **int.test**, works like the others in this series. The regression model is fit to each species data, the distance between the two species vectors (of intercept coefficients in this case) is measured and the observed value is compared to the values that can be obtained by randomly permuting the species' affiliations. Using the data set that you loaded to run **common.slopes.test** and **ont.conv.test**, run the script for the **int.test**. You need to specify the value for *x*, i.e. for the size at which you want the comparison to be done. In principle, that value is zero because that is the value for *x* at the intercept; but you need not use that value to run the test. However, if the slopes really are identical, the value for *x* does not matter because the difference in elevation is constant. This test is done only if the null of the **common.slope.test** is not rejected, and if the *p*-value for the **ont.conv.test** is between 0.025 and 0.975 so, unlike the input sizes for **ont.conv.test**, the input size for **int.test** does not matter.

```
Results3=int.test<-(size,shape,0,species,nperm=1000).
```

het() (R)

The code for testing the hypotheses of ontogenetic scaling and heterochrony via size-shape dissociation, provided by Sylvain Gerber, tests for overlapping trajectories in size-shape (**TfH1()**) and for overlapping trajectories in shape space (**TfH2()**). **TfH1()** and **TfH2()** are the two main functions and there are two additional ones, **reg()** and **smd()**. **reg()** performs the regression (which can be linear, quadratic or higher order; as written, it is quadratic). **smd()** is the function used in **TfH2()** to estimate the distance of each shape to the nearest point on the regression curve. The tests work by fitting separate regressions for each of two species and summing the squared residuals. The residuals are randomized and the sum of squares is calculated; the proportion of times that the sum of squared randomized residuals is lower than the observed sum of squares gives the *p*-value. The test for size-shape dissociation uses the second function (and **smd()**).

The input data consist of a column of sizes (X), a matrix of shape data, with each individual in a row (Y) and a column of species' affiliations (gp), the codes for the groups (coded as 1 and 2). So, you would load your data, assign the column of centroid sizes to the vector X, the shape data to matrix Y and the column of species' codes to the vector (gp). So, if you have loaded file with species' codes in the first column and centroid size in the second column and shape coordinates in the remaining columsn, the first step is to enter:

```
gp=myData[,1]
X = myData[,2]
Y=myData[,-(1:2)]
```

Because the code is written for species labeled as either 1 or 2, the group codes must be these two numbers. If you have multiple species and the codes range from 1 to the number of species, then you need either to renumber the species codes for each pairwise comparison or, for each pairwise comparison you do, alter the code for all the functions wherever you see either X[which(gp==1)]),Y[which(gp==1),] or [which(gp==2)]),Y[which(gp==2),]). Change 1 and 2 to the values you have given the codes for that pair of species; make sure to change them in both functions that require the **gp** variable.

To run the functions **TfH1()** or **TfH2()**,enter

```
Results1 <-TfH1(X,Y,gp)
Results2 <-TfH2(X,Y,gp)
Results1
Results2
```

Then find something else to do for a while because **TfH2()** is very slow.

The ontogeny of disparity

Three programs can analyze the change in disparity from one developmental stage to another, two of which are suitable only for pairwise comparisons between means. The one that can analyze any number of species is **DisparityBox7 (IMP**), introduced above. The two that can analyze pairs of species are **TwoGroup7** (IMP), also introduced above, and **ont.conv.test()** (**R**), another program by Piras and colleages.

DisparityBox7

DisparityBox7 can calculate the disparity of juveniles of multiple species, and of the adults of those species. Thus, it is possible to compare the disparities of the juveniles to that of the adults. As noted above, **DisparityBox7** provides a confidence interval for disparity rather than a statistical test of the difference between disparities.

To use **DisparityBox7** to compare the disparities of two age-classes you will do two analyses. The first will measure the disparity of one age class, the second will measure the disparity of the other. If your data are from discrete age classes, you can load the data for multiple species either by loading each file one at a time or by using the chain file. The chain file lists the file names (plus extensions) for all the data files that you want to load. Remember that this file must give the full path name for all the files on the list unless they are in the same folder as **DisparityBox7**.

If you have continuous ontogenetic series, you will need to prepare the **Log Size Target List**, which **DisparityBox7** will use to predict the expected shape at that size for that species. This is done by fitting a regression model to the data, calculating the expected shapes at the sizes listed in the **Log Size Target List** adding the randomly drawn residuals to each expected value for shape and refitting the model and finally producing the standardized data. This entire process is redone at each iteration of the bootstrap; based on these size-standardized values, **DisparityBox7** calculates disparity (i.e. the variance computed over group means). The outcome is a measure of the observed disparity and a confidence interval that incorporates the uncertainty of the regression.

Once you have loaded your data, and the **Log Size Target List** if you are using it, select **Find Grand Consensus Mean(Groups),** then go to the **Multi-Group Analysis** menu and either select **Bootstrap Geometric Disparity** (for data from discrete size or age classes) or **Bootstrap Size-Corrected Disparity** (for data from continuously sampled ontogenetic series).

The results for the first analysis, e.g. of the juveniles, will appear in the **Results Window**; these can be saved to a file (go to the **File** menu and select **Save Results Box**).

After completing (and saving) the results for the first analysis, do the same for the other age class. You can save those results to the same file (using the **Append Results Box to File**).

TwoGroup7

To use T**woGroup7** to analyze the change in disparity, you would load the data set for the juvenile age classes of two species, and then the data for the adults (or other age class) of two species. **TwoGroup7** would then determine whether the first distance differs from the second. To use **TwoGroup7** for this purpose, use **Load Dataset 1** to load the juveniles of one species, and use **Load Dataset 2** to load the second file of juveniles. Then go to the **File** pull-down menu and use **Load Group 3** and **Load Group 4** to load the two samples of adults. Now go to the **More Stats** pull-down menu and select **Bootstrap Distances 1+2 vs 3** + **4**. When the iterations are completed, the results will show up in the **Results** window. These results will include the Procrustes distance between means 1 and 2, the 95% range and standard error for the distance. Scrolling down you will see the same information for the distance between means 3 and 4. After that you will come to the 95% range for the difference between the distances. If this includes zero, then the distances are not significantly different. If both ends of the range are positive numbers, the first distance is significantly longer than the second; conversely, if both ends of the range are negative numbers, the second distance is larger than the first.

ont.conv.test()

ont.conv.test is another program by Piras and colleagues, so the same input data can be used in this test as in the **common.slope.test**. What this test does is to measure the distance between a pair of species at a selected minimum size and at a selected maximum size and to test the significance of that distance by the same randomization procedure used in the **common.slope.test**. As noted above, the *p*-value is the proportion of times that the difference computed from the randomized data equals or exceeds the observed one; if the difference between the permuted data always exceeds the observed distance, the *p*-value will be 1.00.

To run the analysis, you need to enter more information than you did to run the **common.slope.test** because **ont.conv.test** requires the two sizes at which to do the comparisons. A single value for the minimum and a single value for the maximum sizes is required by the program for all comparisons between all pairs of species. For example, we could enter:

```
Result3<-ont.conv.test(size,shape,3.5,5.7,species,nperm=1000)
Result3
```

The disparity of ontogeny

To analyze the disparity of ontogeny, by which we mean the disparity of the ontogenetic trajectories themselves, you need a file containing these trajectories. Each one should be a row in the data matrix. You can get the coefficients of the trajectory from **Regress7a,** or from **lm()** in **R** or from any other statistical program. **Regress7a** gives allometric coefficients of partial warps (because the program uses partial warps in the statistical analysis). For the trajectories to be comparable, meaning that the first partial warp for one species is the same variable as the first partial warp from another, you need to use the same reference form when computing the partial warps. To do that, use the **Load Reference** button right below the **Use GLS Reference**. This step will ensure that the partial warps for all species are based on the same reference form. You can save either the (**Normalized) Growth Vector** or the **Deformation Vector**; the difference between these two vectors is that the growth vector is normalized to unit length whereas the deformation vector is not. The first of these vectors measures only the direction of ontogenetic change, the second incorporates information about the magnitude of the change as well. To save one (or both) go to the **File** menu and select the vector of your choice. Delete the centroid size column before loading it into the program that will do the analysis of disparity.

To get the allometric coefficients in **R**, you would run the regression, then save the coefficients. To do the regression of shape coordinates contained in a matrix that we will call **data** on centroid size, contained in the vector, **size**, enter:

```
coeff.data <- lm(data~size)$coeff
```

The allometric coefficients (and intercept terms) will be in **coeff.data**. These coefficients are the allometric coefficients for landmarks, which will make it easier to interpret the results of the analyses, such as the principal components of the ontogenetic trajectories. The axes will be dimensions of variation of the ontogenetic coefficients of landmarks rather than of partial warps. So you can see which combination of coefficients vary the most among species and draw the picture of the dimension of greatest variation in allometry, as a deviation from the mean allometry.

This is only an ordination method, not a statistical analysis of disparity. The disparity of the trajectories would be measured as the variance of the coefficients but there is no software that can test the null hypothesis of no difference in disparity, taking the uncertainty of the regression into account.

Standardization

Size standardization (or "size correction) can be done in an IMP program, **Standard7**, or in **R**. **MorphoJ** also has the ability to save residuals (or predicted values) from one analysis to be used in another. See the section in the help file for the **Covariation** menu: **Residuals/Predicted Values From Other Regression.**

Standard7

Standard7 works somewhat differently from most other programs in the IMP series in that it requires that you actively accept the defaults. As usual, you load the data in the standard X1,Y1…CS format. The data set may be landmarks or partial warp scores. The data will be plotted in the visualization window (but when they are partial warp scores, the plot will be meaningless). You need to say whether the independent variable is in the last column of your data file (where CS is usually located) or instead is contained in another file. To say that it is in the last column, select **Use x** = **CS** (the independent variable need not be centroid size; selecting this option means that the variable is located where CS usually is). Alternatively, you can select **Load x-List**, which allows you to input a file containing the values of the independent variable. This file must be a single column of data, with one entry per specimen (in the same order as the specimens are in the input data file). The values must be numerical, no letters or formatting codes can be read (although they can be included in the file by placing them to the right of a % sign).

An error message will appear if the number of specimens and number of values in the independent variable do not match. Also, you will get an error message if there is a hard-return after the last value in the independent variable list (if you get an error message that doesn't make sense, check for this possibility).

You can choose to regress on the values in the last column of your data file (or on the input independent variable file) or you can transform it, either to ln(x) or log(x). Once you have made your choice (where you are asked to **Accept: Regression Function**), click **Accept**. The last choice you need to make is the value of the independent variable; the default is the minimum value of *x*, but you can choose the mean or the largest value, or type in one of your choosing. Once you have decided whether to **Accept: Standardize on x** =, click **Accept**.

Clicking on the **Do Regression** button runs the program. You can then show the standardized data (they are plotted in red if you ask to **Show Standardized Data**). As usual, you can copy the image to the clipboard (with or without axes on the plot) using the **Copy Image to Clipboard** button, and you can save the standardized data in the X1, Y1…CS file format by asking to **Save Standardized Data**.

Before loading your next file, click on **Clear Data**.

Size-standardization in R

Size standardization in **R** can be done by doing the regression, using the fitted values to get the expected value for shape at the observed maximum and minimum values for size, or by selecting two sizes, the minimum and maximum values. Both procedures work by predicting shape at the chosen size, and adding the residuals to it. If you want to standardize the data to both the minimum and maximum size (or to two selected sizes), then it is convenient to make a column of labels to indicate which are the rows standardized to the minimum versus maximum size. This script makes columns of labels ("1" for the residuals added to the fitted value at the minimum size, "2" for the residuals added to the fitted value at the maximum observed size). There is a second script, slightly modified, that uses two input values for size rather than the observed minimum and maximum sizes.

To use the minimum and maximum observed sizes:

```
data=read.table(file.choose())
shape=as.matrix(data[,-(1)])
size=as.vector(dat[,1])
LCS=log(size)
X=LCS
p=ncol(shape)
N=nrow(shape)

#Do the regression
model1=lm(shape~X)

fit.val=model1$fitted.values
#Get the expected values at minimum and maximum sizes
shapeMin<-fit.val[which(X==min(X)),]
shapeMax<-fit.val[which(X==max(X)),]
minShape=rep(1,N)%*%t(shapeMin)
maxShape=rep(1,N)%*%t(shapeMax)

model1.res<-model1$res


#Add residuals to minimum and and max values
model1.res=as.matrix(model1.res)
shapeMin.stand<-model1.res+minShape
shapeMax.stand<-model1.res+maxShape
```

```
shape.stand=rbind(shapeMin.stand,shapeMax.stand)
labelMin=rep(1,N)
labelMax=rep(2,N)
level=c(labelMin,labelMax)
stand=as.matrix((cbind(level,shape.stand)))
```

To use two input sizes instead, input the desired minimum and maximum sizes

```
data=read.table(file.choose())
shape=as.matrix(data[,-(1)])
size=as.vector(dat[,1])
LCS=log(size)
X=LCS
p=ncol(shape)
N=nrow(shape)
#input desired minimum and maximum sizes here
minX =
maxX =

minXc=c(1, minX)
maxXc=c(1,maxX)

#Do the regression
model1=lm(shape~X)

shapeMin=as.numeric(crossprod(coef(model1),minXc))
shapeMax=as.numeric(crossprod(coef(model1),maxXc))
model1.res<-as.matrix(model1$res)

#Add residuals to minimum and and max values

shapeMin.stand<-model1.res+minShape
shapeMax.stand<-model1.res+maxShape
minShape=rep(1,N)%*%t(shapeMin)
maxShape=rep(1,N)%*%t(shapeMax)
shape.stand=rbind(shapeMin.stand,shapeMax.stand)

#Make the labels
labelMin=rep(1,N)
labelMax=rep(2,N)
level=c(labelMin,labelMax)
stand=as.matrix((cbind(level,shape.stand)))
```

Literature Cited

Adams, D. C., and Collyer, M. L. (2009) A general framework for the analysis of phenotypic trajectories in evolutionary studies. *Evolution* **63:** 1143-1154.

Collyer, M. L. and Adams, D. C. (2007) Analysis of two-state multivariate phenotypic changes in ecological studies. *Ecology* **88:** 683-692.

Gerber, S. and Hopkins, M. J. (2011) Mosaic heterochrony and evolutionary modularity: The trilobite genus *Zacanthopsis* as a case study. *Evolution* **65:** 3241-3252.

Li, S. (2011) Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics & Statistics* **4:** 66-70.

Mitteroecker, P., Gunz, P. and Bookstein, F. L. (2005) Heterochrony and geometric morphometrics: a comparison of cranial growth in *Pan paniscus* versus *Pan troglodytes*. *Evolution & Development* **7:** 244-258.

Piras, P., Salvi, D., Ferrar, S., Maiorino, L., Delfino, M., Pedde, L. and Kotsakis, T. (2011) The role of post-natal ontogeny in the evolution of phenotypic diversity in *Podarcis* lizards. *Journal of Evolutionary Biology* **24:** 2705-2720.

12

Evolutionary Developmental Biology (2): Variational Properties

This chapter discusses the software for implementing the methods for analyzing the variational properties discussed in the textbook: (1) plasticity; (2) canalization; (3) developmental stability, (4) the relationship between plasticity, canalization and developmental stability; and (5) modularity. Most methods needed for studies of plasticity and developmental stability were covered in Chapters 9 and 11. In Chapter 9, we presented methods for comparing means across treatments and for analyzing fluctuating asymmetry. In Chapter 11, we presented methods for comparing the magnitude and directions of ontogenetic trajectories, using methods that apply to any kind of trajectory including plastic responses to environmental variables. This chapter thus focuses on software for comparing the levels and structures of variance and fluctuating asymmetry (FA), and for analyzing modularity.

Comparing levels of variance

Levels of variance within samples of 2D data can be compared in **DisparityBox (IMP)** and by an **R** script that is based on one written by Nathan Young (**Bootstrap Variance.R**). **Bootstrap Variance.R** can also compare the variances of 3D data.

DisparityBox7 (2D)

The first step is to load your data sets, in IMP format. Each data set should be in a separate file. Load the first one by clicking on the **Load Data Set** button, find the first file, then click that button again to load the next file. Estimates of variance can be based either on the input data or on size-standardized data from which the variation explained by size has been removed. That standardization is done by performing a regression, computing the mean shape at a requested size, and adding the residuals from the regression to that mean size. To use this procedure, you need to prepare a list of the desired sizes for the means, the **target size list**. This is one column of numbers, the desired size for each sample, ordered in the same sequence as the samples were loaded (i.e. the first size on the list is the target size for the first group loaded). The sizes should be in units of log transformed centroid size (to either base *e* or base 10). For example, the following list says to standardize the first group to 3.1 LCS, the second to 2.3 LCS, and the third to 3.4 LCS

```
3.1
2.3
3.4
```

Load the list either by clicking on the Load **Log Size Targets** button, or go to the **File** menu on the toolbar and select the **Load Log Size Target** option.

As each file is loaded, its superimposed landmarks will appear in the small visualization window on the upper right. To get a list of the files that you loaded, click on the **List Loaded Sets** button. To save that list, go to the **File** menu on the toolbar and click on **Save Results Box**.

Before doing an analysis, you need to superimpose the two samples. There are two options for calculating the mean (they produce identical results when samples sizes are equal in all populations). Select either **Find Grand Consensus Mean (Specimens)** or **Find Grand Consensus Mean (Groups)**. The first uses an unweighted mean (so it is influenced by the sample sizes of the group); the second uses the mean of the means.

To compare the variances, go to the **Multi-Group** menu, and select **Bootstrap Pairwise Difference in Disparity** and/or **Permute Pairwise Difference**. For both, the default number of bootstraps and permutations is 200. **Bootstrap Pairwise Difference in Disparity** bootstraps each sample, computes the variance within each one at each iteration of the bootstrap and calculates the difference between the two variances. The null hypothesis is that the difference is zero. The last line of the output, which appears in the Results Window, shows the 95% confidence interval for the difference in the variances. If it includes zero, the null hypothesis cannot be rejected. This shows what the results look like: The output refers to the (Foote) disparity because the program also analyzes disparity following the methods devised by Foote.

```
For Group 1 from File  one Proc.txt
-------------------

 The distance-based (Foote) disparity was
 : 0.00131845

The 95 percent confidence interval for the disparity over the bootstraps was
 :  0.00100224 to 0.00144076

 The Standard Error for the disparity over the bootstraps was
 :  0.00011148
For Group 2 from File  ten proc.txt
-------------------

 The distance-based (Foote) disparity was
 : 0.00071903

 The 95 percent confidence interval for the disparity over the bootstraps was
 :  0.00053245 to 0.00081266

 The Standard Error for the disparity over the bootstraps was
 :  0.00006998

The observed disparity of 1 minus Variance of 2 is: 0.000599

 The 95 percent confidence interval for the difference in disparity over the
bootstraps was
 :  0.00029267 to 0.00079602
```

The analysis can also be done by permutations. **DisparityBox** reports the number of permutations in which the magnitude of the observed difference was exceeded:

```
For Group 1 from File  one Proc.txt
-------------------

 The distance-based (Foote) disparity was
 : 0.00131845
For Group 2 from File  ten proc.txt
-------------------

 The distance-based (Foote) disparity was
 : 0.00071903

The observed disparity of 1 minus Variance of 2 is : 0.000599

 The 95 percent upper bound on the permutation generated difference in
disparities is
 : 0.00031916
```

The magnitude of the observed difference was exceeded by 0 of 200 permutations.

To save the results to the same file in which you listed the loaded data sets, go to the **File** pull-down menu and select **Append Results Box to File**, then select the file in which you saved the list of loaded files. The program will warn you that the file will be overwritten, but it won't be. Or you can simply copy the results from the **Results Window** and then paste that into a text file.

If you wish to analyze the variance of traditional morphometric measurements, you need to load a measurement protocol. This consists of a three-column list; the first column is the number of the measurement, the second is the number of the landmark that will serve as one endpoint of the measurement, the third is the number of the landmark that serves as the other endpoint. For example, the measurement protocol for lengths measured between landmarks 1 and 7, between landmarks 2 and 4, and between landmarks 4 and 5 is:

```
1 1 7
2 2 4
3 4 5
```

(This is the same protocol used in the program **TradMorphGen** which is one of the modules of **CoordGen7**.) Load the protocol either by clicking on the **Load Length Protocol** button, or by going to the File menu on the toolbar and selecting the **Load Length Protocol** option.

BootstrapVariance.R (2- and 3D)

Bootstrap Variance reads a file of variables, calculates the observed variance within each of two samples, bootstraps each sample (the default is 1000 times) and calculates the variance within each sample at each iteration and the difference between the variances. Results include the observed variances; the *p*-value, which is the number of times that the variance of the first sample exceeds that of the second; and q, the quantiles corresponding to the probabilities that you request (the defaults are 0.05 and 0.95).

To run **BootstrapVariance()** on shape data, the input files should contain only the variables whose variance you want to estimate. Thus, if loading an IMP file, remove the centroid size column before running **BootstrapVariance();** similarly, if loading a file formatted for **MorphoJ**, remove the Identifier column. Open the script and select the three functions (**varT()**, **bootstrapVar()** and **diffVar()**). Run them. Then load your data, select and run the rest of the script. The last four lines show the results: the variance of each sample, *p*-value and quantiles.

Comparing the structures of (co)variance

Covariance structures are usually compared by a matrix correlation. To compare the covariance matrices of symmetry and FA, you can use **MorphoJ** or **Mace**, another program in the Morphospice series

(Marquez, 2012b). Both programs implement the **Mantel test**. The Mantel test, adapted to morphometric data, is also available in **R** using one of Claude's functions, **mantel.t2()**. It is important to remember that the null hypothesis is of no similarity between the matrices (i.e. that they are no more similar than expected by chance). If the null hypothesis is rejected, that does not mean that the matrices are highly similar, it just means that they are more similar than two random matrices are expected to be. The alternative null hypotheses, of matrix equality or proportionality, may be more relevant to your study. This test is rarely done, but if your hypothesis predicts that covariance matrices are highly stable, this may be the more appropriate test to do. We first discuss programs that use the Mantel test, then one that tests the null hypothesis that the matrices are drawn from the same population.

In **MorphoJ**, go to the **Variation** menu and select **Matrix Correlation**. In the pop-up window, name the analysis, and select the two matrices that you want to compare. Decide whether you want to include the diagonal of the matrix (i.e. the variances) in the comparison. The default is to include them. Decide how many permutations you want to do; the default is 10,000. You can also choose whether to permute landmarks or coordinates; the default is landmarks (not *x*-,*y*- coordinates separately). Then click **Execute**. You will see a scatterplot that shows the variances/covariances of one data set plotted against the corresponding variances/covariances of the other data set. Sometimes you will see one or a small number of very high values that look like gross outliers in the plot. This is more common when you include the diagonal because variances are typically much higher than covariances. If one or a few landmarks have exceptionally high variances, they will be outliers on the plot. To see the statistical result, go to the **Results** tab.

To run **Mace** (either 2- or 3D versions) click on the **Load File** button to load your files. You can load files of coordinates in IMP format (or without centroid size in the last column), paired variables (right and left) such as partial warp scores, traditional morphometric variables or covariance matrices. If you want to assess the correlation between the symmetric and FA components of object symmetry, you will need to load the files (saved in **Sage**) that contain only the paired landmarks. Select the kind of file that you want to load then select the first file. Repeat the process until you have loaded all the files that you want to compare. **Mace** will do pairwise comparisons between all loaded files. Then decide whether you want to **include the diagonal** of the matrix (i.e. the variances) in the comparison (the button for this choice is to the right of the **Load File** button). The default is to not include them. Then select the tests that you want to do. You can select **Bootstrap Repeatability** to estimate the correlation between a matrix and resampled versions of itself as a means of quantifying the reliability of the correlation, **Bootstrap 95% CI** to put confidence intervals on the correlation between two matrices, and/or the **Mantel test**. When you've selected the tests that you want to do, click on **Compute Correlations**. You can save the results to a file using the **Save Results** button

Mantel.t2() (R, 2- and 3D)

Mantel.t2() depends on three other functions from Claude's Morphometrics in **R**: **lower.triang()**, **perm.rowscols**. and **mantrstat().** If you did not source the code in **functionsR()**, find these plus **Mantel.t2** and read all of them into **R**. The arguments for the function include the two matrices that you want to compare, the number of coordinates per landmarks (i.e. 2 for 2D data, 3 for 3D data), the desired number of permutations and whether you want a graph (TRUE or FALSE). The function returns the matrix correlation and the *p*-value.

To run **Mantel.t2()**, enter:

```
Result.cor <- Mantel.t2(Matrix1,Matrix2, 2, nperm = 10000)
```

MatrixEquality() (R, 2- and 3D)

MatrixEquality(), written by Annat Haber, tests the null hypothesis that two matrices differ by no more than expected by chance, i.e. they were drawn from the same population. The null hypothesis is thus that the correlation between the matrices is 1.0 rather than 0.0. The method works by drawing two matrices, at random, from a multivariate normal distribution. The structure of the observed matrix is imposed on the random matrices, which differ from each other only by random, normally distributed noise. The matrix correlation is computed between this pair of matrices, and the process of drawing the matrices at random, imposing the observed correlation structure of them, and estimating the matrix correlation is repeated 1000 times. This generates a distribution of matrix correlations under the null hypothesis that the two original matrices differ only by random, normally distributed noise. The program returns the observed matrix correlation, the repeatabilities of the two observed matrices (the correlation between the matrix and boostrapped versions of itself) and the probability that the correlation between the two observed matrices is no lower than expected under the null. There are two *p*-values, one based on the first observed matrix, the other on the second.

We made one modification of the program so that it would run when there are more variables than individuals. The program uses the function **rmvnom()**, which provides a random number generator for the multivariate distribution with a mean equal to one and a covariance matrix sigma. The default method for the function is **eigen** which requires having more individuals than variables. We therefore changed two lines, from the default **eigen** to **svd**

```
X1 <- rmvnorm(N, sigma=R,method="svd")
X2 <- rmvnorm(N, sigma=R,method="svd")
```

To run **MatrixEquality()**, you need two covariance matrices. You can generate them from your files of superimposed shape coordinates. If your files are in IMP format, remove the last (centroid size) column. If they are in MorphoJ format, remove the first (ID) column. You will also need to input the samples sizes for each sample, so if you do not recall that, add the final two lines that output N1 and N2.

```
data1<-read.table(file.choose())
data1<-read.table(file.choose())
cov1<-cov(data1)
cov2<-cov(data2)
N1<-nrow(data1)
N2<-nrow(data1)
```

Open **Matrix.Equality()** in the script window, select all of it and run it. Then, at the command prompt type:

```
res<-bootMR1(cov1,cov2,N1,N2)
res
```

The first line sends the output to **res** and the second shows **res**.

Comparing levels and structure of FA

FA can be analyzed using the same procedures that we discussed above for comparing the level and structure of variances. To apply them to FA you need a file of the right-left differences in shape coordinates for each individual. That file can be obtained from **Sage**, which adds those right-left differences to the symmetric mean shape. That file can be analyzed just like the files of the averaged right-left sides (i.e. the symmetric component). Studies of FA typically report the results in terms of an average deviation rather than as a variance so you can modify the **BootstrapVariance** script to take the square root of the sum of the eigenvalues. You can also use those coordinates to compare the structure of FA across populations.

Assessing the relationship between plasticity, canalization and developmental stability

To compare the relationships between these properties, you need measures of each. Often, canalization and developmental stability are measured at the individual level, by computing the magnitude of each individual's deviation from the mean or, in the case of developmental stability, each individual's deviation from symmetry. An individual's deviation from the mean, in units of Procrustes distance, can be measured either by calculating the Procrustes distance of each individual from the mean directly, or by saving the D vs CS output from **Regress7A**, using the GLS mean as the reference. The output file has two columns, one is the Procrustes distance of an individual from the reference (here, the mean shape), the other is centroid size. If the input file contains the coordinates for FA added to the mean symmetric shape, the output file will contain the deviations of each individual from that shape.

To get the D vs CS file, open **Regress7A,** and **select CS** (untransformed CS) then **Load Data**. Use the **GLS reference** option. Then go to the **File** menu and select **Save Distance vs CS/LCS** (you can also use the **Save Distance vs CS or LCS** button on the interface, located in the lavender box on the far left).

Correlating scores

When you have the scores for each individual, you can measure the correlation between their deviations from the mean shape and their deviations from symmetry. This can be done in any statistical program.

Comparing covariance matrices

To compare the covariance matrices of FA and symmetric variation of shapes that exhibit matching symmetry, you can use the same programs that we discussed above for comparing covariance matrices of individual variation. Additionally, you can also get the correlation between symmetric (Individual) variation and FA while doing the analysis of FA in **Sage**. Go to the green **Specify Correlations** box below the yellow MANOVA box, check the small box in the left hand corner to tell **Sage** that you want to compute the matrix correlations, then select the components whose correlations you want to test from the pull-down menu. When you click on **Run analyses**, the analysis of FA will include tests of the requested matrix correlations.

To compare the covariance matrices of FA and symmetric variation for shapes that exhibit object symmetry, you will need files that contain only the unpaired landmarks. If you did not save those when you exited **Sage**, then you will need to rerun the analysis to get them, and while rerunning the analysis you can include the matrix correlation as part of the re-analysis. It is worth having those files so that you can compare multiple samples - just because you can get the correlation between FA and symmetric covariance structures while running **Sage** does not lessen the value of saving the files of coordinates.

Modularity

This section discusses four programs for analyses of modularity. **MorphoJ** (Klingenberg, 2011) implements what we called the "Minimum inter-modular covariance method" in the textbook. Briefly, this method computes a measure of the covariance between postulated modules relative to covariance within them (**RV** or $\mathbf{RV_M}$), and compares that value to the distribution of **RV** or $\mathbf{RV_M}$ values obtained by randomly partitioning a configuration of landmarks into the same number of "modules" as predicted by the hypothesis, with those modules having the same number of landmarks as the modules predicted by the hypothesis (Klingenberg, 2009, 2011). **Mint**, another program in the Morphospice series (Marquez, 2012b), implements what we called the "Minimum deviance method" in the textbook. Briefly, this method assesses the relative fit of multiple models to the data by the γ^* statistic, a measure of the deviance between model and data, adjusted for the number of free parameters (Marquez, 2008; Parsons et al., 2012). Two programs implement the method that we called the "Distance-matrix method" in the textbook. Briefly, this measures correlations between putative modules by calculating the pairwise Procrustes distances between all specimens for each partition, and computing the matrix correlation between all pairs of partitions; the resultant correlation matrix can be analyzed by the methods applied to correlation matrices obtained from traditional morphometric data (Monteiro et al., 2005; Monteiro & Nogueira, 2009). One program that implements this method is **Coriandis** (Marquez, 2012a), another program in the Morphospice series. A second, **doItAllForMantel()**, written by Adam Rountrey, runs in **R**; and offers two methods, one of which separately superimposes the partitions, the other of which simultaneously superimposes them (and does not resuperimpose them after partitioning).

In our descriptions of the programs, we first explain how the hypotheses are expressed, then how to obtain the tests.

Minimum inter-modular covariance method, MorphoJ (2- and 3D)

Specifying the hypothesis of modularity

Assuming that you already have loaded your data set into **MorphoJ**, your first step is to specify a hypothesis of modularity. To do that, go to the **Project Tree** and select a data set and, if you have not already done so, go to the **Preliminaries** menu and select **Generate Covariance Matrix**. Then go to the **Covariation** menu, select **Modularity: Evaluate Hypothesis**. A window will open that has three tabs, one of which is **Select subsets of landmarks**. When you choose that tab, you will see a picture of your landmarks and to the right of the graphical window, you will see a small box below the label **Number of subsets**. That box contains a pull-down menu; select the number of modules contained in your hypothesis. In response to that number, a series of colored boxes will appear in the graphical window. Each color corresponds to one of the modules and, as you assign landmarks to one of them, the landmarks will be color coded. To assign landmarks to the modules, first select a subset number (save #1 for last because its landmarks will not change color when you assign them to a subset). After you select the subset number, assign the landmarks belonging to that subset to it. Holding down the Shift-key lets you assign a series of landmarks that are successively numbered to the same subset. Holding down the Control-key lets you assign multiple landmarks that are not successively numbered to the same subset. If you forget a landmark (or more) that should have been put in that subset, you can easily add them by just selecting the number of that subset and assigning the landmarks to it. Repeat the process of selecting a subset number and assigning landmarks to that subset until all your hypothesized modules are specified. Every landmark must be assigned to a subset (the default subset is #1) and no landmark can belong to more than one subset.

If you wish to test your hypothesis by considering only spatially contiguous random partitions, you need to provide **MorphoJ** with the information about the spatial contiguity of the landmarks. A preliminary hypothesis is indicated by the gray lines connecting landmarks in the modularity windows. For mouse and squirrel jaws, landmarks on different mandibular processes are guessed to be adjacent. Go to the **Modify adjacency graph** tab to add and/or delete links. One way to add links is by selecting the numbers of the two landmarks that ought to be linked; you do that within the two small boxes on the right side of the interface. Select the two landmarks and click on **Link Landmarks**. A similar approach can be used to delete links; in the large box that lists all the links, find one(s) that you want to delete and click on **Delete Link**. The alternative method for adding links is to draw one between two landmarks on the picture. You can select a link to delete by drawing a line between the endpoints of the link (as if you were adding it); that will then be selected in the **Delete Link** box and you can accept the deletion.

This process of specifying the hypothesis can take a long time, but once you have successfully done it you can copy the landmark partitions and adjacency graph from one data set to another. To copy the partitioning and adjacency graph to another data set, select the hypothesis of modularity in the **Project Tree**, right click to **Display Graphs**, then select **Hypothesis**. You can copy the partitioning by right-clicking and selecting **Copy Information**. Then, go to the **Select subsets of landmarks** tab for the other data set, and right click to **Paste Partition**; go to the **Modify adjacency graph** to **Paste Connectivity**. When you are satisfied with your partitions and adjacency graph, you click **Accept**.

You will get the window to **Start analysis**. First, **name** your analysis (this is advisable because you may test multiple hypotheses, perhaps for multiple data sets, before you are done). Then, if you want to constrain the random partitions to be spatially contiguous, click the check box. You will also be asked whether you want a full enumeration of all partitions (or to all spatially contiguous ones) or instead you want a number of random partitions (select the number of partitions you want). The program is fast enough that you can ask for at least 10000 random partitions (the default number). If you have a fairly small number of landmarks, you could ask for a full enumeration.

You will get both graphical and numerical results. One graphical result, on the tab for the **RV** or **Multi-set RV** (RV_M) coefficient, is a histogram of the distribution of the **RV** or **RV_M,** with a large red arrow showing where your hypothesis fits within that distribution. You will also see a tab, **Minimum Covariation** which, not surprisingly, shows the partitions of landmarks with the lowest covariation between blocks (the remaining tab shows your hypothesis, the **Hypothesis** tab). The analysis will produce graphical output in three tabs inside the **Graphics** tab. On the **Results** tab, there is summary information about the hypothesis (the number of subsets and the landmarks within them) and the **RV** or **RV_M** is given and, below that, the number and proportion of alternative partitionings that produced a smaller **RV** or **RV_M**. There is also a description of the hypothesis that produced the lowest covariation among all the hypotheses considered (both the landmarks within the partitions and the **RV** or **RV_M** for the hypothesis).

You can copy the information about the landmark partitions to the clipboard and paste it into a text-editor if you want to reproduce that same partitioning in one of the other programs.

Minimum deviance method, Mint (2D)

Specifying the models

In **Mint**, as in **MorphoJ**, every landmark must be assigned to a subset (i.e. a module). However, in **Mint**, a landmark can be assigned to more than one module, and a landmark can be a module in its own right. Proposing modules that consist of a single landmark does present a problem when combining the modules of different models to generate new models and when using **Part-Whole PLS** (which will not run when the modules contain fewer than three landmarks). But you may have sampled some modules so sparsely that you need to make one landmark a module in its own right. Models that contain single-landmark modules can be assessed for their goodness-of-fit.

The models can be prepared in a text-editor or spreadsheet and loaded into **Mint** or built and edited within **Mint**. If preparing a file that specifies a model in a text-editor or spreadsheet, make two columns; the column on the left contains an integer for the number of the subset, an identifier for the module, with as many rows as there are landmarks per module. The column on the right contains the number of the landmarks that belong to that module. Both columns must be numbered successively from lowest to highest number. So, if there are 10 landmarks in two modules, and each contains five landmarks, the hypothesis that landmarks, 2, 3, 4, 5 and 10, belong to one module and 1, 6, 7, 8 and 9 comprise another looks like this:

```
1 2
1 3
1 4
1 5
1 10
2 1
2 6
2 7
2 8
2 9
```

If your landmarks were digitized in order, so that anatomically adjacent landmarks are also numerically sequential, the model is easy to write out. But, if anatomically adjacent landmarks are not numerically sequential, it may be easiest to write out the landmarks in anatomical order and then sort the file so that the modules are ordered sequentially and the landmarks within modules are also ordered sequentially.

If you want to build your models within **Mint**, or if you have built them in a spreadsheet and now want to test them, you need to load your data into **Mint**. To load the data, make sure that **Landmark data** is checked. Then click the **Load Data** button and load your data. The data-file formats that **Mint** can read include IMP format, with or without a ruler, that same format without centroid size in the last column, and (equivalently) **MorphoJ** format without the identifier in the first column. **Mint** can also read tps files, but only if the control lines are limited to LM=, IMAGE=, ID= and SCALE= in the file. **Mint** crashed when a file included the CURVES = and POINTS= lines.

If you are loading models as well as data, you can load each model one by one or use a batch load. The advantage of a batch load is that you will have a file that lists the files loaded and the sequence in which they were loaded. As a result, you will have a record of which model is model1; when you return to the analysis days or months later and can no longer recall the order in which you loaded the files, the batch file list will remind you. For that reason, give each model an informative (but brief) name. You want a brief name because typing errors will prevent loading the files. To prepare the batch file, write out the lists of model file names in a text file, using an editor that will not turn quotes into smart quotes or add other characters that might prevent the file names from being recognized. If the models are not located within the same folder as **Mint**, you need to include the whole path name. This is part of the batch file included with **Mint.**

```
jaws.model.01.txt
jaws.model.02.txt
jaws.model.03.txt
jaws.model.04.txt
jaws.model.05.txt
```

To load the batch file, first check the **Batch load** box next to the **Load Models** button. Then load the batch file. If not using a batch file, load each model one by one.

After loading your data, you can **build** models within **Mint** (or **edit** the models that you loaded). To build the model, go to the **Tools** pane on the right side of the interface and select **Model building**. A list of landmarks will appear in the box labeled **Landmarks**. To build a model, you select the landmarks that belong to the first module then go to the **Tasks** pane (on the right, below the **Tools** pane) and press **Select module**. In the black rectangle below the graphic window, **Mint** tells you what you need to do next. Alternatively, you can draw a polygon around the module. This can be rather challenging. When you click on the picture, you will get what look like cross-hairs, and you can draw a line between two landmarks, and then extend it, forming a triangle including the third landmark, and extend that, making a polygon. At the end, you right click and the enclosed landmarks will be selected on the **Landmarks** box. The instructions within the black rectangle guide you throughout this process.

To edit the models in **Mint**, go to the **Tools** pane and select **Modeling Editing**, then select the model to edit on the **Model** list (which has replaced the **Landmark** list) and click on **Select Model** on the **Task** pane. Then select the module to edit on the **Modules** list (which has replaced the **Models** list) and click on **Select MODULE** on the **Tasks** pane. At that point, the list of landmarks appears in the **Landmarks** list. The landmarks that belong to the selected module are highlighted. You can then either select the landmarks from that list or draw the polygon. The instructions within the black rectangle will again guide you throughout this process.

Testing models

When you have all your models loaded, built and/or edited, you can test the models. The default test statistic is the scaled **Gamma***; there are two other statistics available in **Mint**, but neither is scaled for the number of fixed parameters. One is the matrix correlation; the other is the angle between subspaces. The matrix correlation, in this case, is between the model and the data. The angle between subspaces is another measure of matrix similarity that was covered in the first edition of our textbook, but this statistic is not often used and we did not include it in this edition (For an explanation of this test statistics, see Zelditch et al., 2006). The choices for the test statistic, and the options for the jackknife tests, are on the yellow bar above the graphics window.

After selecting the test statistic, select **Model Testing** from the **Tools** pane and select the models that you want to test from the **Models** list on the left. You will notice that there is one more on the list than the number you loaded - the first model on the list is the null model (of no integration - a model that places each landmark in its own module). Select the models that you want to test, and click on **Select models** on the **Tasks** pane and then select **Run test** from the **Task** pane.

After the models are tested, the list of models in the **Model** window will be sorted from best- to worst-fitting, as judged by the scaled value of the test statistic. The **Results** window will contain the results for each model, giving the number of the model that is currently selected, the standard γ^* value (or other test statistic) for the model, and the *p*-value for the null hypothesis. In the case of the γ^* statistic, the null is that the data are no more different from the model than expected by chance (based on a Wishart/Monte Carlo test with the number of replicates you chose); and the arrow in the histogram shows the position of the model with respect to the null distribution for the hypothesis. You can switch between a graphical display of the model and the histogram using the **Display model/Ranked model** button on the graphical window; what the button says depends on whether you are looking at the model or the histogram.

After these tests are done, you can assess the jackknife support for the ranking of the models and produce a confidence interval for the test statistic. To do that, first make sure that the number of replicates requested in the **No. Jackknife reps** box is larger than zero (which is the default value). Enter your desired number of replicates, the proportion of the sample that you wish to drop and the % confidence interval that you want to produce. If you want to put a confidence interval on the test statistic, you should use a large number of replicates (e.g. 1000); the default value for the percentage of individuals to drop is 10%, and the default confidence interval is 95%. After choosing the values that you want, on the **Task** pane, select **Jackknife support**.

Combining modules from different models

When you have completed the analysis of the models you loaded or built, you can combine modules to create new models. These can be tested along with the originally loaded ones. This can be an extremely informative procedure because you may find that all the good-fitting models, both those you derived from biological theory and those produced by combining modules, have features in common that distinguish them from the poor-fitting models. You may well find many models that fit exceptionally poorly, and the features that make them fit so poorly can also be illuminating.

To combine models, select the **Model Testing** tool on the **Tool** pane, and select two or more modules from the list (first selecting the models in the **Models** window, and then clicking on **Select Models** button on the **Task** pane). It can take a very long time to search for all unique combinations of modules for even just four models so consider how many you want to combine. After the combinations are made, you can see the models by clicking on the **Display Models** button in the graphics window. From a list that might number well over 1000, you can select the ones that you would like to include in the next test. Some of the models will contain only one module, others will have extensive overlap between modules. You could select them all or an array of models that you pick as examples of conflicting hypotheses.

You can save several output files (click on **Save Outputs** and a list will appear whose options depend on what you have done). If you have only done the initial tests, the only options are to **Save the Test Statistic and P-values** and the **% Confidence Limits of Null Distribution**. If you have also estimated jackknife support, you can save the support values for all ranks, or save the model ranks for all jackknife repeats plus the confidence intervals for the test statistics and if you have combined models, you can save current models.

If you save the **Test Statistics And P-Values**, you will get a matrix with as many rows as the number of models you selected to test. The models are listed in the order that they appear in your batch file. The first column contains the test statistic in raw form, the second contains the test statistic standardized by its maximum possible value, the third contains the *p*-value unless the test statistic is γ^*, in which case that is in the third column and the *p*-value is in the fourth.

If you save the **X% Confidence Limits**, where **X** equals the percentage of confidence that you specified for the jackknife support, you will get a matrix of three columns. The rows are in the same order as the output above. The first column contains the % confidence for the interval, the second and third contain the lower and upper limits for the raw test statistic, respectively. These are computed from the percentile of the distribution of values obtained from the Monte Carlo replicates. **The Save Jackknife Support Values For All Ranks** produces a file with as many rows as the models tested and the columns are the ranks for each model for each replicate of the jackknife. **The Save Model Ranks for All Jackknife Reps** produces a matrix in which the columns contain the ranks for one replicate of the jackknife test. The **Save Confidence Intervals For All Test Statistics** contains the lower and upper confidence intervals for the goodness-of-fit statistic, estimated from the jackknife replicates.

The **Save Current Models** will produce files for each of the models that are loaded, including the ones that you created by combining the models..

Part-Whole PLS

To visualize the relationships between a single part and the whole morphology, select **Part-whole PLS** on the **Tools** task pane. You then select the model on the **Models** menu, and click on **Select Model**, after which you select the part on the **Parts** menu and click on **Select Parts**. Obviously, the landmarks of the part will covary with those same landmarks within the whole, but the objective is to determine whether they also covary with other landmarks. The first singular axis will appear in the graphics window - the others can be selected from the **PLS axes** menu. You can show the picture of the axes either as vectors or as deformed grids (there are options for the displays of the deformed grids on the **Plot** controls at the top left of the interface, above the **Load Data** button). Numerical results are shown in the **Results** window, including the % covariance explained by the selected axis and the correlation between the scores of the two blocks on each axis. You can do a permutation test to assess the significance of the covariance explained and the correlation.

After running the **Part-Whole PLS** procedure, you will have additional outputs to save. The **Save Output** options now include **Save Singular Values and Proportions of Captured Covariance** and **Save PLS Angles and Correlations**. The **Singular Values and Proportions of Captured Covariance** file contains the singular values in the first column, the proportion of the covariance explained by that axis, permutation *p*-values (if permutations were done) in the third column, and *p*-values for the null hypothesis that the cumulative covariance accounted for by successive axes is no greater than expected by chance. The **PLS Angles and Correlations** file contains the angle, in degrees, between each pair of part and whole axes, in the first column, and the correlation between them in the second and, if permutation tests were done, the *p*-values for the null hypothesis that the correlation is no larger than expected by chance. Interpreting these results can be challenging when the covariance explained by the first axis is relatively low - given that the landmarks within the part are all also within the whole, we would expect the first axis to explain a very high proportion of the covariance between the blocks. That is not always the case, however; see the **Mint** manual for an interesting discussion of the interpretation of such results.

One of the main uses of the **Part-Whole PLS** procedure is to refine the models. So after running the analysis you can return to the **Model Building** or **Model Editing Tools** (or to a text-editor) and use what you've learned from the analysis of the **combined modules** plus **Part-Whole PLS** to devise new models for the next round of testing.

Distance-matrix method, Coriandis (2D)

Coriandis measures correlations between shapes of modules or parts of modules that are separately superimposed. The pairwise Procrustes distances between individuals within each partition are computed after superimposing the individual partitions. If you want to use a simultaneous superimposition approach, such as those implemented by the other two methods, see **doItAllForMantel()**, below.

To run **Coriandis**, you need to subdivide the configuration of landmarks into separate files, each one corresponding to a module or part of a module. This subdivision can be done using the programs discussed in context of obtaining the two blocks for **PLS** in Chapter 7 (Partial Least Squares). Two programs in the IMP series can do this partitioning, producing data files in the format read by **Coriandis**. One is **VisProto7**, the other is **LMEdit7**. (Briefly: in **VisProto7**, from the **Additional Editors** menu, select **Subset Generator** and then type (or paste) a list of the landmarks that belong to a single subset into the **Current Subset List** window. To check that you've selected the right landmarks for this subset, click **Show All Subset Points,** and to save the subset click on **Save this Subset**).

To load your partitioned data files into **Coriandis**, you can either load one file at a time or use a batch loader. If you load one data set at a time, click on the **Landmark data** button on the upper right of the interface. When the window opens select your file and the window will reappear so you can load your next one. When you are done loading them all, click **Cancel**. The batch file for **Coriandis** differs from the one used by **Mint** because, in **Coriandis**, you need to say what type of data you are loading. The three types are: landmark data (**L**), distance matrix (**D**) or other (**O**). The batch file should include file the name of the files to load; followed by a comma and then by TYPE=. If the data files are not in the same folder as **Coriandis**, the full path name is needed. An example of part of a **Coriandis** batch file looks like this:

```
C:\landmarks\dataset1.dat,TYPE=L
C:\distances\dataset2.dat,TYPE=D
```

The next step is to tell **Coriandis** whether your data are grouped. If you are analyzing modularity within a population, click on **No Groups** just below the **Batch Load** buttons. Then select the analysis that you want to do. The default selections are not for studies of modularity within groups so you will need to deselect these and select **Compare distances using Mantel test**. The default number of permutations is 100. That should be increased to at least 1000 if you want a statistical test of the correlations. If your sole objective is to get a correlation matrix for further analyses, the number of permutations can be left at 100. Then click the green **Run** button at the bottom right of the interface. When the analysis is done, you will see a picture in the graphics window, but the default plot is not a picture of the results of your analysis. To see those, go to the **What to Show?** window above the graphics window, and select **Matrix correlations**. The correlation coefficients and *p*-values are displayed in the **Results** window and you can select which to show using the pull-down menu in the **What to Show**? menu. To save the results, go to the **Saves** menu on the upper left corner of the interface. The results of your analysis are at the bottom of the list: **Save matrix correlations** and, if desired, **Save Mantel test P-values**.

Distance-matrix method, doItAllForMantel(), R (2D, modifiable to 3D)

To run **doItAllForMantel(),** a script by Adam Rountrey, you need two files: your data file (in IMP format, with or without centroid size in the last column) and a partition file, which is the file that specifies which landmark belongs to which partition. The first column of the partition file names the partition and the successive columns list the landmarks that belong to that partition. This should be a comma-delimited file; part of a partition file looks like this:

```
AntIncisor,2,16,17,18,19,20,1,31,32,33,34
PostIncisor,21,22,23,24,25,35,36,37
```

doItAllForMantel() does both a simultaneous and a separate superimposition distance-matrix analysis, using the **procGPA** function in the **shapes** package to superimpose the data. The first procedure is therefore to convert the loaded IMP file to the **shapes** data-file format. The data are subdivided into the partitions specified by your partition file. The partitioning will be displayed in the Graphics window, with landmarks color coded by partition. The pairwise distances are calculated between individuals within each partition file and the matrix correlations between all pairs of partition are calculated and tested for statistical significance by the Mantel test. The partitions are converted back into IMP format in case you want to save those.

To run **doItAllForMantel()** and save the results into **Modularity**, enter

```
Modularity=doItAllForMantel(file.choose(),file.choose())
```

A window will appear to let you navigate to your data file, and it will appear again to let you navigate to your partition file. When the program finishes running, the results will be contained in **Modularity**. The results include the matrix correlations for the simultaneous analysis, which will be in **Modularity$notSepManS**, and the *p*-values for these correlations, which will be in **Modularity$notSepManP**. Similarly, the matrix correlations for the simultaneously superimposed results will be in **Modularity$notSepManS** and the *p*-values for those correlations will be in **Modularity$notSepManP**.

Once you have the correlation matrix, you can use test hypotheses that predict the correlations between the parts using any of the methods applied to traditional morphometric data, including a Mantel test between the observed and expected correlation matrices. Because these are correlations between shapes of parts, not coordinates of landmarks, you can use a standard Mantel test (not one adapted to landmarks) such as **mantel()** in the **vegan** package. You can also use graphical modeling, for which there is freely available software, including **MIM** (http://www.hypergraph.dk/) whose manual is contained in

12. Evolutionary Developmental Biology (2): Variational Properties

Edward’s (2000) Introduction to Graphical Modeling and **ggm**, (Marchetti and Mathias Drton, 2010) a graphical modeling package in **R**.

Literature Cited

Edwards, D. (2000) *Introduction to Graphical Modelling,* 2nd ed. Springer-Verlag, New York.

Klingenberg, C. P. (2009) Morphometric integration and modularity in configurations of landmarks: tools for evaluating a priori hypotheses. *Evolution & Development* **11**: 405-421.

Klingenberg, C. P. (2011) MorphoJ: an integrated software package for geometric morphometrics. *Molecular Ecology Resources* **11**: 353-357.

Marchetti, G. M. & Mathias Drton, M. (2010) Graphical Gaussian Models.

Marquez, E. (2012a) Coriandis. University of Michigan, Ann Arbor.

Marquez, E. (2012b) Mint. University of Michigan, Ann Arbor.

Marquez, E. J. (2008) A statistical framework for testing modularity in multidimensional data. *Evolution* **62**: 2688-2708.

Monteiro, L. R., Bonato, V. & dos Reis, S. F. (2005) Evolutionary integration and morphological diversification in complex morphological structures: mandible shape divergence in spiny rats (Rodentia, Echimyidae). *Evolution & Development* **7**: 429-439.

Monteiro, L. R. & Nogueira, M. R. (2009) Adaptive radiations, ecological specialization, and the evolutionary integration of complex morphological structures. *Evolution* **64**: 724-743.

Parsons, K. J., Márquez, E. J. & Albertson, R. C. (2012) Constraint and opportunity: the genetic basis and evolution of modularity in the cichlid mandible. *American Naturalist* **179**: 64-78.

Zelditch, M. L., Mezey, J., Sheets, H. D., Lundrigan, B. L. & Garland, T. (2006) Developmental regulation of skull morphology II: ontogenetic dynamics of covariance. *Evolution & Development* **8**: 46-60.

13

Systematics

This chapter is essentially empty because the methods that would be described here are covered in other chapters or have not yet been developed.

In the textbook, we identified three broad categories of questions that systematists have tried to address using morphometric methods: taxonomic, phylogenetic and evolutionary. Taxonomic questions concern the discrimination of taxa, phylogenetic questions concern the inference of phylogenetic relationships (evolutionary or phylogenetic trees), and evolutionary questions concern the inference of historical evolutionary transformations of morphology along the branches of the phylogenetic tree.

For taxonomic questions, there is a well developed toolkit of methods consistent with the theory of shape that is the foundation of geometric morphometrics. This tool kit includes the description of axes of variation and differentiation and tests of the statistical significance of hypothesized groupings. Most of these methods were described at length in previous chapters (especially 6 and 8) and will not be reprised here. The exceptions are methods that combine size and shape information in discrimination tests. To the extent that size aids discrimination of the groups in question, these methods may improve on discrimination based solely on shape. Readers interested in these methods are referred to the next chapter, 14.

In contrast to the myriad tools for taxonomic discrimination, we find that the toolkit for phylogenetic inference using shapes is still, disappointingly, empty. To be sure, there is a wealth of techniques for inferring phylogenetic relationships from morphology and other traits; what we lack are methods for turning shape data into variables that can be used in phylogenetic inference without violating the principles of geometric morphometrics.

Methods for reconstructing histories of shape evolution were discussed in Chapter 10. As we noted there, these methods are most fully developed for testing inferences under the model of Brownian motion drift, but alternatives are available.

14

Forensic Applications of Geometric Morphometrics

This chapter discusses the software used in the forensic applications of geometric morphometrics presented in the textbook. These methods are primarily concerned with discrimination of groups and testing group membership – questions that also are relevant outside of forensics (as in the diagnosis of taxonomic or ecological groups). One difference between methods described here and those described in earlier chapters (especially 6 and 10) is inclusion of size data in the analysis. Methods that include size data are the focus of the first section of this chapter. Another departure from more familiar discrimination analyses is the test for matches, meaning for shapes that cannot be distinguished from one another within the margin of measurement error. In the second section, we present a method of performing that analysis. Finally, we expand on our previous discussion of the k-means test (Chapter 10), focusing on its use for testing group memberships.

One technique discussed in the textbook is the use of relative eigenanalysis to develop a discrimination function and a log-likehood test based on that function. As discussed in the textbook, this approach has been used to detect evidence of Fetal Alcohol Spectrum Disorders (Bookstein et al., 2001, 2002a, 2002b; Bookstein and Kowell, 2010). The analysis is not particularly difficult to program in **R** or Matlab, however, there are several steps requiring decisions between several alternatives. Among the more important decisions to be made are the choice of likelihood ratio to use and what statistical distribution should be fit to the data. Perhaps because of this complexity, there are no readily available programs or **R** scripts implementing this method.

Analyses including Size and Shape

There are two approaches here. One is to use a standard Procrustes superimposition, and append centroid size or log centroid size as an additional data column. The other, called Procrustes-SP (Size Preserving) methods, uses only rotation and translation in the superimposition, leaving the size of all specimens unchanged. The latter Size-Preserving method produces configurations in Form space – the space of configurations that centered and rotated to minimize the (squared) distances between corresponding landmarks, differing only in size and shape. The principal difference between the two approaches is whether the size information is collected in one variable or distributed across all landmark coordinates. The degrees of freedom will be 2*k*-3 for both data sets, but the number of variables will be 2*k* or 2*k*+1.

Canonical Variates Analysis of "size and shape" with CVAGen7

Recall that data files in IMP format (such as that output from **CoordGen7a**) have centroid size as the last column in the data matrix. To include log centroid size when you run the CVA in **CVAGen7a**, simply select the **Include CS in CVA** button on the bottom center area of the window *before* loading data. This analysis includes a standard Procrustes superimposition with the shape rescaled to have unit centroid size; the original centroid size (actually, log centroid size) was included as an additional variable in the computation of the CVs.

One limitation is that **CVAGen7b** cannot perform the PCA reduction of the size plus shape data before the CVA. If you need the PCA reduction, you can perform the PCA in **PCAGen**. Then, copy the result to a spreadsheet and delete the unwanted PCs (remembering to retain the centroid sizes). Finally, perform the CVA in a conventional statistical package (because **CVAGen7a** cannot accept PC scores as input).

Producing size-preserving superimpositions in CoordGen7

The IMP program **CoordGen7a** can be used to perform Procrustes-SP superimpositions, which can then be exported for use in a variety of other applications. (This is also a way to get IMP coordinates scaled to their original sizes so that the correct centroid size will be computed when they are read into **MorphoJ**.)

If you previously saved a data file in IMP format using **CoordGen7a**, you can load that file back into **CoordGen** using the **Load BC File** option. If you have a tps file with scale factor or ruler, load that by selecting the appropriate option. After loading the data, click the **Fixed Scale Tools** menu to obtain the following options.

- **Show Procrustes (Size Preserving)** – plots your data in Procrustes SP superimposition.
- **Save Procrustes SP data** – saves the data in this superimposition for analysis elsewhere. (This can also be found in the **File Options** menu.)
- **Save Procrustes SP mean** – saves the iteratively estimated Procrustes SP mean of your data set.
- **Variance/Scatter in Procrustes SP** – writes several measures to the **Auxillary Box**, including the variance about the mean specimen. Variance is measured as the summed squared distances of individuals from the mean, divided by (N-1) for N specimens (where an individual's distance is the summed squared distances of corresponding points). These are Procrustes distances in the sense that a minimization has been applied, but they are not distances in either commonly used shape space. The minimum and maximum observed distances of specimens from the mean are also shown, as is the mean distance of all specimens, and the RMS scatter, which is the square root of the variance described above.

Looking for Matching Shapes

In looking for matching shapes, one first needs to determine a cutoff distance (in Procrustes and/or Procrustes SP distances). If one accepts the definition of a match put forward in the text:

"If the observed difference between two measured specimens is no larger than the difference observed in repeated measurements of a single specimen, then there is no evidence that the two measured specimens are different, and thus they may be said to match"

then the process of looking for matches has two steps:

(1) Determine the maximal distance expected when a single specimen is measured repeatedly

(2) Search through a database for all pairs of specimens with the matching distance determined in step 1, or compare a given specimen to all other specimens to look for a match.

The more difficult step is determining the expected value for the difference in repeated measures of single specimen. In a series of experiments, Bush and colleagues observed that roughly 93-96% of all repeated measurements of a single specimen fell within twice the root-mean-square (RMS) scatter of the repeated measurements about the mean of those measurements. This RMS scatter is simply the square root of the variance, measured using summed squared Procrustes (or Procrustes SP) distances about the mean. The RMS scatter may be estimated by repeating digitizing a specimen, and using **CoordGen7a** to estimate the RMS scatter as discussed above. One might also estimate this matching range as the 95th percentile range of all inter-specimen distances among repeated measurements obtained from **CoordGen7a**, or perhaps even the maximum observed difference between repeated measurements of the same specimen. This matching distance will typically depend on the measurement method, the type of measurements used and operator capabilities, so it will need to be determined for each experimental protocol, operator and group of specimens. In any case, it will require measuring representative specimens a large number of times.

Testing for Matches with MatchMaker (IMP)

Once the RMS scatter level or other criterion for the matching distance is obtained, testing a set of collected specimens for matches is quite simple, the **MatchMaker** program in the IMP series may be used to search for matches.

- **Load Input File** to read in a dataset in IMP format.
- **Show Histogram of All Specimens** to compute a histogram of all pairwise distances between specimens. This is a required step in the operation of this program, and will show you a list of the closest matches in the data set. When this is completed, you can save a listing of all pairwise distances in the data set using the **Save Distances for All Specimens** button.
- To determine the number of specimens in the sample within your selected matching distance of each other, go down to the **How Many Specimens Below** box, enter the matching distance in the space to the right of the button, and then click on the button. The box below the histogram will tell you how many pairs meet that criterion, and how many different specimens (labeled **dentitions**) are in at least one of those pairs. To perform the analysis with Procrustes SP data, click that button before requesting the search.
- To illustrate the shape difference between a pair of specimens (e.g. the two closest), enter their numbers in the box below the **Plot Listed Specimens** button, then click that button.
- To test whether a single specimen that was not included in the set matches a specimen in the set, you must have that specimen in a file by itself, also in IMP format. Click the **Load Target Specimen** button and locate that file. After both files are loaded, click **Match Target in Data Set**. You can also save a list of the distances from this target specimen to all other specimens in the data set using the **Save Distances for Target Specimen** button.

- You can also **Load a Second Data Set** of more than one specimen, using the button in the lower right corner. Clicking the button below this, **Find matches between 1, 2**, will start the search for matches between the first and second data set. In effect you are now looking for matches to more than one target specimen at a time.

Testing Group Membership

As discussed previously, ordination methods like PCA and CVA can be used to evaluate whether *a priori* groups can be discriminated using the available measurements. In general, group membership is taken as given, so the statistical evaluation focuses on the ability of the available data to replicate correctly the original classification (which can then be used as measure of the discriminant function's ability to classify unknowns). An alternative approach might be to use cluster analysis to build groups based on some measure of similarity or distance, then evaluate whether the clusters are congruent with the hypothesized; however, cluster analyses are fraught with numerous risks, including dubious joining algorithms and forced hierarchical structure. In many cases, all you really want to know are the number of groups and who belongs to them, not whether there is hierarchical structure within or between these groups. This much narrower question can be addressed by **k-means** analysis.

In brief, k-means analysis begins by dividing the data set into *k* groups (where *k* is an integer), by randomly assigning individuals to one of those groups. Group means (centers) are calculated and then each specimen is re-assigned to the nearest center, based on the chosen distance metric. New group centers are calculated and assignments are re-evaluated until group memberships are stable. The results include a listing of the final group memberships, which can be compared to your *a priori* classification to compute an assignment rate.

k-means analysis in R

The **kmeans** function is within the **stats** package in **R**, which is often installed by default.

Shape data files in IMP format may be loaded directly into **R** with the **read.table** function, since each specimen is on a single row of the file, with landmarks in Cartesian coordinate pairs (*x* followed by *y*) with the centroid size appearing in the last column. Use **CoordGen** to save your data, consisting of *k* groups of specimens, loaded in some known order, just as you would when preparing data and a group list for IMP for PCAGen and/or CVAGen, in your desired superimposition (using Partial Procrustes data superimposed on a GLS mean of all specimens would result in an analysis in the linear tangent space, and is probably the first superimposition to try). Then load it into **R** using:

```
X<-read.table(file.choose())
X<-as.matrix(X)
n<-ncol(X)
k<-3 # or whatever number of groups you want to use in the test
y<- kmeans(X[  , 1:(n-1)], k) # to include size, use X[,1:n]
```

The function will return:

- **y$cluster** – a vector of integer values (1 to *k*) indicating which cluster (group) each individual is assigned to. You can then determine from these codes if specimens are assigned to the same groups as you would have assigned them *a priori*, allowing you to determine an assignment rate. If you have a group code list prepared, you can compare the *a priori* group code list to the cluster list produced by **kmeans** by loading both into Excel for comparison purposes.
- **y$centers** – the mean of each cluster on each variable. If your data were shape coordinates, plot these to show the shapes of each cluster. One might compare these to the means of each *a priori* group.
- **y$totss**, **y$withinss**, **y$tot.withinss**, **y$betweenss** – which are the total sum of squares (ss), the within-group ss for each group, the total within group ss (sum of with-group ss) and the between-group ss. The ratio of between-group ss to the total within-group ss is a measure of the explanatory power of the groupings.
- **y$size** – this is a listing of the number of specimens within each group.

The k-means function in **R** does offer several different k-means grouping algorithms (see the **R** manual for details), it may be advisable to experiment with these different algorithms to determine if one is preferable given your data set, or if the performance is similar for all approaches. Also, results can depend on the initial random assignment, so it would be a good idea to repeat the analysis a few times, especially if there are no clear gaps between groups.

Literature Cited

Bookstein, F. L. & Kowell, A. P. (2010) Bringing morphometrics into the fetal alcohol report: Statistical language for the forensic neurologist or psychiatrist. *Journal of Psychiatry and Law* **38**: 449-472.

Bookstein, F. L., Sampson, P. D., Streissguth, A. P. & Connor, P. D. (2001) Geometric morphometrics of Corpus Callosum and subcortical structures in the fetal-alcohol-affected brain. *Teratology* **64**: 4-32.

Bookstein, F. L., Sampson, P. D., Connor, P. D. & Streissguth, A. P. (2002a) Midline Corpus Callosum is a neuroanatomical focus of fetal alcohol damage. *Anatomical Record* **269**: 169-174.

Bookstein, F. L., Streissguth, A. P., Sampson, P. D., Connor, P. D. & Barr, H. M. (2002b) Corpus Callosum shape and neurophyschological deficits in adult males with heavy fetal alcohol exposure. *NeuroImage* **15**: 233-251.