Appendices

Cambridge University Press, the publisher of this book, has graciously agreed to post diverse online resources that will aid you in making use of the methods that this book surveys or the examples it exploits to show the methods hard at work. These resources come without any warranty, so to speak. In particular, regarding the software components, I don't assert that they are reliable in all applications, only that they are indeed the software I used for producing the majority of the analyses and figures of shape data in the main text. There certainly exist better (i.e., more professionally compiled) software libraries for this purpose – the list includes software from Ian Dryden, Stanislav Katina, Dean Adams, and several others – but access to an algorithmically accurate library does not automatically enable you to transform its computations into effective vehicles for your findings.

Even in the context of measurements that are extents (lengths and areas) (Chapters 2 and 3), where all packages get almost exactly the same answers for invocations of standard linear models, there is still very little assistance offered in wrapping the resulting computations for export. (Ask yourself, for instance, if your favorite package has a dialogue box that would let you set up Figure 2.40 by clicks.) By definition, all of my examples in this book count as "publishable imagery," in that they have all been published right here. You may find a wide range of these fully explicit examples, some from distances or areas and some from shape coordinates, useful to examine, imitate, or improve prior to being woven into an ongoing argument of your own in a dissertation, scientific manuscript, or colloquium talk.

The contents of these appendixes are laid out in four categories that match the four subdirectories of the associated Cambridge website. Appendix A lists the thirteen data sets that drove most of the empirical examples in the book. Appendix B presents and explains the four main shape space tools that produce the vectors of loadings and scores driving my examples in Chapter 5: `Fproc.new` (Procrustes shape coordinates in two dimensions), `Fpw.from.proc` (partial warp loadings and scores), `Fdrawtps` (the thin-plate spline grid for arbitrary pairs of landmark configurations), and `Frelwarps` (for exploring patterns of shape variation at large scale). Appendix C surveys a range of useful additional tools for diagramming patterns or explanations. The final Appendix D is a bulky compilation of my actual Splus scripts (mostly usable in R as well) for most

of the book's figures that convey actual findings (as distinct from pedagogical diagrams, proofs of theorems, and the like).

It is inevitable that there will be errors here, as well as inefficiencies of loop structure or name assignments. Please call any anomalies that you uncover to my attention at `flb@stat.washington.edu` or `fred.bookstein@univie.ac.at`.

Appendix A Data Sets

The data sets used for most of this book's examples are listed explicitly in the online resource associated with this publication. Some are mine while others are my selections from larger online resources posted by others as acknowledged in the entries that follow. The relevant section of `S.reading.data.sets` includes the few Splus statements needed to import each of these data sets in turn to that computing environment.

A.1 Vilmann Neurocranial Octagons

The file `vilmann.data.set` in the online resource extracts 144 lines from the original 164-line file printed *in extenso* in Bookstein (1991, Appendix A.4.5). The restriction here is limited to the 18 animals with complete data at all ages. This classic resource plays many different roles in my text, beginning with the first figure of Chapter 1 and ending with the final figure of Chapter 5. File format: 144 lines with animal number, age in days, and then eight (x, y) coordinate pairs (in a system with Bregma at $(0, 0)$ and Lambda on the negative x-axis). All configurations are in the same unit of measurement, but I seem to have no record of what that unit was: maybe μ? In the relevant part of the `S.reading.data.set` file, these data are read as Cartesian coordinate pairs and immediately converted to complex numbers, rendered as Procrustes shape coordinates, and submitted to partial warp and relative warp analysis.

A.2 Twenty-Nine Anthropoid Skulls

The file `jhe03.data.set` in the online resource includes 20 landmark points and 74 semilandmarks for each of 29 anthropoid skulls, as originally published in Gunz et al. (2003). Like the Vilmann data set, these midsagittal cranial configurations appear in this text in many different contexts, beginning in Chapter 1. The appropriate paragraph of `S.reading.data.sets` imports all 94 points as Cartesian coordinate pairs, converts them to the complex numbers that the main Splus routines prefer, assigns names to the points and the specimens, and then extracts subsets and derives structures

useful in the demonstrations to follow. File format: 29 chunks of 94 (x, y) pairs, one pair per line, totalling 2,726 lines. The raw data were integers, but the 74 semilandmarks are results of a matrix manipulation, hence decimals.

A.3 The "Baby Brain" Data Set

The file `baby.brain.data.set` in the online resource includes four landmark points from synthetic midsagittal ultrasound images of the infant corpus callosum for 23 babies whose mothers reported having exposed them to high levels of alcohol before birth along with 21 others. These data are exploited here in Example 3.9, Figures 3.46 et seqq. The ultrasound source images, as reported in Bookstein et al. (2007), are not to any consistent scale, and so they have already been converted to Procrustes shape coordinates. File format: an exposure code (1 = unexposed to alcohol before birth, 2 = exposed) followed by four points in the form of complex numbers.

A.4 Human Adult Female Footprints

The file `domjanic.data.set` in the online resource lists 36 semilandmarks from a synthetic section of three-dimensional surface scans of both feet for 79 adult Croatian women as analyzed in Bookstein and Domjanić (2015). The word `domjanic` stands for Dr. Jacqueline Domjanić of Zagreb, Croatia, part of whose dissertation data set this was. The data will be processed by the sliding algorithm `Ftpsoutline` and used here in Figure 5.82. Format: each 36-gon is spread over four lines of eight complex numbers and a fifth line of four. Left and (mirrored) right foot are paired.

A.5 Marcus–Hingst–Zaher Mammalian Calvarial Midlines

The file `marcus57.projected.coordinates` in the online resource lists 13 midsagittal landmark locations for representative specimens of 57 mammalian taxa. The larger landmark set from which this extracts the unpaired points was reported in Marcus, Hingst-Zaher, and Zaher (2000). These data provide the analysis displayed in Figures 5.59–5.60 here (although 5.60 required additional information, not in this online resource, about the evolutionary history of these specimens). File format: 57 lines of thirteen complex numbers each, derived as projections onto the first two principal axes of the original 3D scatter of these points. The `S.reading data.sets` file includes text strings for specimen names (vernacular, not Linnaean) and long landmark descriptions.

A.6 Isthmus Transects

The file `isthmus.data.set` in the online resource lists 45 Cartesian two-vectors representing transects of the isthmus of the callosal midcurve at its shortest average

section for three groups of Seattle adult males imaged by brain MRI in 1998. Exposure is coded in an integer from 0 to 3: code 3 for 15 subjects previously diagnosed with Fetal Alcohol Syndrome, code 2 for 15 with Fetal Alcohol Effects, and code 1 for the 15 whose mothers, interviewed in 1974, assured us they had not exposed their fetus to prenatal alcohol. The 46th line, assigned to group zero, is for a client (whose initials were not "XX") from my forensic practice. The larger semilandmark set from which this is a selection is not yet released for publication. These isthmus transect data appear in Figures 3.43 et seqq.

A.7 United States Death Rates, 2003

Files `life.table.males.2003.csv` and `life.table.females.2003.csv` in the online resource were downloaded from Arias (2006). Their role in this book is to supply the closest fit to a straight line of any empirical example (Figure 3.21). File format: introductory text followed by 100 lines each of seven entries. The example in the text involves only the third column of these data sets.

A.8 Wright's Leghorn Chickens

The file `wright.chickens.data.set` in the online resource lists ten measures, of which we use the primary six, for 276 leghorn chickens that were used in Sewall Wright's famous analysis of "general" and "special" size factors driving Figure 2.7 and 2.31. This file was downloaded from an online resource at Jim Rohlf's `morphmet` web site.

A.9 Bumpus's Sparrows

The file `bumpus.data.set` in the online resource lists nine morphometric measures for 136 sparrows collected after a snowstorm in Providence, Rhode Island, in 1898. The examples in the book use either all the adult males (N = 59) or the 36 surviving adult males. These data appear in Sections 3.1 and 3.6. The original data were published in Bumpus (1898); my listing is a version online at the Field Museum, Chicago.

My variant `Fcdf.pair` of Splus's built-in cumulative distribution function tool `cdf` is used to emulate Bumpus's original observations of group differences variable by variable. The code for this routine is included in the online resource.

A.10 Berkeley Guidance Study Data

The file `berkeley.guidance.data.set` here is copied from the online resource for Weisberg (2005). Weisberg, in turn, extracted these from the full listing published in Tuddenham and Snyder (1954). These data, my favorite example of a hypothesis-rich

longitudinal study of extents (as distinct from shape information), are used extensively in Sections 3.3 and 3.4.

A.11 Tetrahymena Data Set

The file `tetrahymena.csv` was originally posted by Andersen and Skovgaard in the online resource for Andersen and Skovgaard (2010). There are two measures for 51 experimental runs of a study of the protist *Tetrahymena* having two groups: 32 with high glucose and 19 others. These data appear in this text in Section 3.3. File format: 51 lines of three semicolon-separated entries: glucose group (1 = high, 0 = low), concentration, and mean diameter of the experimental samples.

A.12 Birthweight Prediction Data Set

The contents of the `birthweight.prediction.data.set` file were likewise retrieved from the online resource for Andersen and Skovgaard (2010). This file tabulates three measures for each of 108 newborn infants on whom two diameters were measured prenatally by ultrasound and who then were weighed at birth. The read file generates the appropriate transforms of the ultrasound measures (namely, their cubes). File format: 107 lines of four semicolon-separated integers each. These data drive one of the examples of added-variable plots in Section 3.2.

A.13 A Data Set about Mammalian Brains

I transcribed the four long lines of `neuron.data.set` from the printed table 1 in Herculano-Houzel et al. (2015). These are the data for Figure 3.20c: genus names, taxonomic categories, and representative brain weights and neuron counts for 40 taxa of mammals. File format: four comma-separated lists.

Appendix B Code for the Basic Shape Space Manipulations

This section offers accounts or lightly commented listings of the basic computational tactics that lie at the core of nearly every shape coordinate analysis in this book: `Fproc.new` (for Procrustes shape coordinates and certain derived descriptors), `Fpw.from.proc` (for the uniform component and the partial warps of a two-dimensional shape coordinate data set), `Fdrawtps` (for plotting the thin-plate spline for a pair of landmark configurations in two dimensions), and `Frelwarps` (for computing the *relative warps* of a shape coordinate data set – its principal components with respect to Procrustes distance). All the routines are laid out in full in the corresponding online directory.

B.1 `Fproc.new`

This is the basic routine for making Procrustes shape coordinates out of landmark coordinates in the plane. The version in the online resource is called with one argument that is an $n \times p$ matrix of n p-landmark configurations represented as complex numbers. It returns a structure of six elements: a vector of p complex numbers for the Procrustes mean shape, an $n \times p$ matrix for the coordinates of each specimen after it is fit to the Procrustes mean; another $n \times p$ matrix for the differences of those fitted coordinates from their means (the Procrustes shape coordinates per se); a vector of length n with the Centroid Sizes of the specimens that were divided out in the course of the fitting; another n-vector of the rotations (represented as complex numbers); and finally one more $n \times p$ output matrix for the nonaffine shape coordinates, those resulting from an extended fitting algorithm removing any uniform component of the shape change as well as centering, scaling, and rotating.

`Fproc.new` is a loop, but the basic operation of fitting one configuration of landmarks to another is a finite computation carried out (for 2D landmarks taking the form of complex numbers) by the `Fproc.12` function that applies to an already centered target configuration `c1`. Matching Figure 5.42, this reads, in toto,

```
> Fproc.12
function(c1, c2)
{
    c2 <- c2 - mean(c2)
    c3 <- c2/sqrt(sum(Re(c2 * Conj(c2))))
    c4 <- sum(c1 * Conj(c3))
    c4 * c3
}
```

as you will see nested in the more extended routine later here.

All those `Proc.new` outputs notwithstanding, it is worth displaying the core of the function that realizes the process laid out in Figure 5.45. (The following exposition replaces the names of the variables in the actual Splus functions by the names they should have been assigned instead.) At this stage, each of our n specimens has been converted from $2p$ Cartesian coordinates to p complex numbers, centered at mean $0+0i$, and scaled to Centroid Size 1 (sum of squared distances from this shared centroid). Load an $n \times p$ array `originals` with these original standardized specimens, and allocate an array `iterates` of the same size for the results of the computation to come. Furthermore, there is already some tentative "Procrustes average" shape under consideration (if only the first specimen in the data set); call it `testmean`. Then the basic loop is as follows:

```
for (j in 1:n) {
    iterates[j,]<-iterates[j,]*
          (sum(testmean*Conj(iterates[j,]))/
           sum(iterates[j,]*Conj(iterates[j,])))
```

You recognize this from the code of `Fproc.12` above – it computes the rotation necessary as a complex *regression* of the current iterate on the current testmean, and then produces the next version of the iterate as the predicted value of the vector of landmark locations from that regression. The regression has no constant term because both predictor and predictand have means preset to the complex number $0 + 0i$.. The regression residuals are no longer at Centroid Size 1, so, still in the loop over specimen index `j`, we put them back there:

```
iterates[j,]<-iterates[j,]/
  sqrt(sum(Mod(iterates[j,])**2))
```

and then close the loop over specimens and update the average:

```
}
testmean<-apply(iterates,2,mean)
```

And that is all there is to the basic Procrustes algorithm – this loop over specimens is repeated until convergence, which is invariably rapid. At the end, it is helpful to rotate to principal axes horizontal and vertical (the orientation that makes it easiest to compute uniform components). This entails final processing by a routine that should have been coded as follows:

```
testmean.pc1<-princomp(cbind(Re(testmean),
  Im(testmean)))$loadings[,1]
if (testmean.pc1[1]<0) testmean.pc1<- -testmean.pc1
testmean.turn<-complex(r=testmean.pc1[1],
  i=-testmean.pc1[2])
    testmult<- 1
    if (Re(testmean.turn)<0) testmult<- -1
    if (Re(testmean.turn)<sqrt(0.5)&Im(testmean
                    .turn)>0) testmult<-(0-1i)
    if (Re(testmean.turn)<sqrt(0.5)&Im(testmean
                    .turn)<0) testmult<-(0+1i)
    testmean.turn<-testmean.turn*testmult
testmean<-testmean.turn*testmean
iterates<-testmean.turn*iterates
```

where the indented code checks for alignment with long axis horizontal and no inversion. All the Procrustes shape coordinate scatters in this book were produced by this same routine.

B.2 Fpw.from.proc, FLmatrix

These functions are just software implementations of the corresponding algebraic equations in the main text. Launched with an argument `m0` that is the structure output by the `Fproc.new` routine just reviewed, `Fpw.from.proc` produces three new structures of its own. For a data set of n configurations of p landmarks, these are an $n \times (p-2)$ matrix of *partial warp scores*, a $(p-2)$-vector (whose first entry is zero) of the corresponding *bending energies*, and a $n \times (p-3)$ matrix of the principal warps corresponding to the eigenmodes of nonzero bending in the thin-plate model. Notice the call to built-in routines `eigen` (for principal component analysis) and `solve` (for matrix inversion). If you haven't read Splus or R code before, it helps to know that the `$` notation refers to named subelements of a structure; thus, `m0$mean` is the element named "mean" (a vector of complex numbers) from the structure `m0` produced by the `Fproc.new` procedure (Section B.1). The parameters `alpha` and `gamma` are the α and γ from the extended six-dimensional linear formulas in equations 5.7 of Section 5.4.

```
> Fpw.from.proc
function(m0)
{
    m1<-FLmatrix(m0$mean)
    m2<-solve(m1)
    m3<-eigen(m2[1:length(m0$mean),
      1:length(m0$mean)],symmetric=T)
    b.e.<-c(0,m3$values[(length(m0$mean)-3):1])
    m4<-m0$fits%*%m3$vectors[,(length(m0$mean)-3):1,
      drop = F]
```

```
    alpha<-sum(Re(m0$mean)*Re(m0$mean))
    gamma<-sum(Im(m0$mean)*Im(m0$mean))
    m5<-complex(r=alpha*Im(m0$mean),
      i=-gamma*Re(m0$mean))
    m6<-m0$resids%*%m5/sqrt(alpha*gamma)
    m7<-cbind(m6,m4)
    list(wscores=m7,b.e.=b.e.,prinw=m3$vectors[,
      (length(m0$mean)-3):1,drop=F],mean=m0$mean)
}
```

`Fpw.from.proc`'s output begins with "partial warp zero," the uniform term, which is assigned its own special set of horizontal and vertical scores as per the formulas in Sections 5.4.4 and 5.5.3. The structure produced by this routine also feeds directly into the production of the *bending energy – partial warp variance plot* (Figure 5.92):

```
some.pw<-Fpw.from.proc(some.procrustes.analysis)
plot(log(some.pw$b.e.[-1]),log(apply(Mod(some.
  pw$wscores[,-1]**2),2,mean)))
```

The function `FLmatrix` called by `Fpw.from.proc` is occasionally called elsewhere in these data analyses as well (e.g., in `Ftpsoutlinelmks`). Here it is, exactly matching its formulation as a matrix in equations 5.10–5.11 of Section 5.5 for a calling argument of complex numbers. The `ifelse` is to avoid evaluating $\frac{1}{2}r^2 \log r$ at $r = 0$ (diagonals of the matrix U); instead, this value is simply defined to be zero.

```
> FLmatrix
function(coords)
{
    m1<-rep(complex(r=1,i=0),length(coords))
    m2<-m1%o%coords-coords%o%m1
    m3<-Re(m2*Conj(m2))
    m4<-ifelse(m3>0,0.5*m3*log(m3),0)
    m5<-cbind(1,Re(coords),Im(coords))
    rbind(cbind(m4,m5),cbind(t(m5),matrix(0,3,3)))
}
```

B.3 `Fdrawtps`

This is another routine too verbose to be worth printing in full. It is called by the simple command `Fdrawtps(source,target)` and produces a single image, the deformation of an initially squared grid given by the interpolation of the landmark correspondence between the `source` and the `target`. The actual algebra of the correspondence, a thin-plate spline on the landmarks of the `source`, is that of equation 5.12 in Section 5.5. The coefficients of this expression are produced by the single command right at the beginning of the routine that produces the actual equation of the desired spline:

```
tpscoeff<-complex(r=solve(FLmatrix(source),
  c(Re(target),0,0,0),
    i=solve(FLmatrix(source),c(Im(target),0,0,0))))
```

But it takes another 40 lines of much more tedious code to calibrate, compute, and draw the corresponding transformation grid (a matter of three nested loops, for grid row, grid column, and landmark number). At the core of *this* process is the following fragment for evaluating the thin-plate spline whose $p + 3$ coefficients `tpscoeff` we just computed at a list of points that, from the spline's own point of view, is completely arbitrary (namely, the points of the starting grid superimposed over the `source` configuration). Here is the additional code for that evaluation as applied to any list `domain` of new points in that image plane:

```
affine<-rep(tpscoeff[nland+1],length(domain))+
     tpscoeff[nland+2]*Re(domain)+tpscoeff[nland+3]
     *Im(domain)
kernels<-domain%o%rep(1,nland)
centers<-rep(complex(r=1,i=0),length(domain))
     %o%source
sepmtx<-kernels-centers
terms<-Re(sepmtx*Conj(sepmtx))
terms<-ifelse(terms>0,0.5*terms*log(terms),0)
coeffs<-rep(complex(r=1,i=0),length(domain))
     %o%tpscoeff[1:nland]
partials<-terms*coeffs
nonaff<-apply(partials,1,sum)
affine+nonaff
```

I have saved you from the tedium of reviewing the code that sets up this "domain" and also the code that draws the grid lines connecting up the rows and columns.

Extrapolated plots are easily produced by this same `Fdrawtps` routine. The alpha-fold extrapolation of the map between a `source` configuration and a `target` configuration is drawn by the call `Fdrawtps(source,target +(alpha-1)*(target-source))`.

Variant version: `Fdrawtps.scale`, giving me a little more control over the grid per se.

B.4 `Frelwarps`

This routine, another one too verbose to be worth printing in full in these pages, produces the three basic types of relative warps that help us explore the pattern space of landmark configurations in Procrustes shape space. One set of components ("nonaff") applies to the nonuniform subspace of shape variation, another ("unif") to the uniform subspace. These analyses are commensurate in units of Procrustes distance, and the third version ("full") is their geometric combination. Each set of warps is extracted by marshalling the relevant dimensions of shape space and sending them to the built-in

routine `princomp` for eigenanalysis in the standard way. All vectors of coefficients are reported in two different bases at the same time (the original shape coordinates and either the partial warps [for the full and nonaffine analyses] or the horizontal and vertical uniform components [for the full and the uniform analyses]), along with the specimen-specific scores. It is mostly the redundancy of these six different reporting styles that renders the routine unworthy of reprinting here.

Appendix C Diverse Additional Drawing Tools

This appendix assembles a few other routines useful for measuring or diagramming that I have invoked in various places in the course of producing this book's illustrations.

C.1 `Fplotsquare`

This is legacy code from an era when some built-in plotters did not have an option to set the two axes at the same spacing. It is called with a single vector or matrix of complex numbers as its argument, and resets the plotting character size `mkh` (another argument) to the default before it exits.

```
> Fplotsquare
function(plotlist,xl="",yl="",pch="*",psize=0)
{
    plotrange<-max(diff(range(Re(plotlist))),
      diff(range(Im(plotlist))))
    par(pty="s")
    plot(c(min(Re(plotlist)),min(Re(plotlist))
      +plotrange),c(min(Im(plotlist)),
         min(Im(plotlist))+plotrange),type="n",
           xlab=xl,ylab=yl,pty="s")
    par(mkh = psize)
    points(plotlist,pch=pch)
    par(mkh=0)
}
```

Variant versions: `Fplotsquare.noborder`, `Fplotsquare.smalllabel`, each of which just fiddles a bit with the basic layout.

C.2 Fplotpw, Fplotunif

These routines sets up the plot of a bent partial warp (Fplotpw) or a uniform partial warp 0 (Fplotunif) as a thin-plate spline. The arguments include a partial warp structure (output of the Fpw.from.proc command), a warp number (for Fplotpw), and a two-vector of loadings. Here is the core of the code:

```
> Fplotpw
function(parwarps,whichone,realp=0.1,imagp=0.3,
  xl="",psize=0.04)
{
    parw <- complex(r=realp,i=imagp)
    plotvec <- NULL
    nland <- length(parwarps$mean)
    for (i in 1:nland) {
          ctr<-parwarps$mean[i]
          sep<-parw*parwarps$prinw[i,whichone]
          plotvec<-c(plotvec,ctr+sep)
    }
    Fdrawtps(parwarps$mean,plotvec,xl=xl,psize=psize)
}
```

For the rest of this Splus function, see the listing in the explicit Splus code resource.

C.3 Fplotrw

This routine sets up the plot of a relative warp as a thin-plate spline and also as a displacement diagram. The arguments include a relative warp structure (output of the Frelwarps command), the name of a *set* of warps ("full", "unif", or "nonaff"), an index, and a two-vector of loadings. Assuming all these have been set sensibly, here is the core of the code:

```
    plotvec<-NULL
    nland<-length(relwarps$mean)
    scale<-scale*diff(range(analysis$scores[,
           whichone]))
    for (i in 1:nland) {
          ctr<-relwarps$mean[i]
          sep<-complex(real=analysis$vec.ld
              [2*i-1,whichone],
              imag=analysis$vec.ld[2*i,whichone])
    plotvec<-c(plotvec,ctr+scale*sep)
    }
    Fdrawtps(relwarps$mean,plotvec)
```

This code closely parallels that of Fplotpw. For the explicit Splus function, see the online resource.

Variant versions of the function (that should have been toggle arguments instead): `Fplotrw.nogrid, Fplotrw.noplot.` The original function always produced both kinds of graphical output, in the second of which the displacements at the landmarks are just the elements of the vector `plotvec` itself.

C.4 Ftritensor

This routine, too long to print here (because of all the different combinations of triangle edges that the tensor might find itself intersecting), produced Figure 5.68 in Section 5.5. It should have produced Figure 5.70 as well, but it was not coded yet at the time, so 5.70 was produced inline (see `S.chap.5.5`).

C.5 F.convexhull

This is one single command (inside a deep loop) designed to be executed after the points themselves have been plotted (see, e.g., Figure 3.50). The algorithm is a kludge: draw all the interlandmark segments for which all the other landmarks lie on one side or the other of the segment.

```
> F.convexhull
function(points)
{
    npoints<-length(points)
    for (i1 in 1:(npoints-1)) {
    for (i2 in (i1+1):npoints) {
         if(!(min(Im((points-points[i1])/
                     (points[i2]-points[i1])))*
            max((Im((points-points[i1])/
                 (points[i2]-points[i1])))<-1e-06)){
              lines(points[c(i1,i2)])
}}}}
```

C.6 Fshapecoords

This maneuver, utilized throughout Section 5.2, is just one line when called with a matrix of complex numbers as argument:

```
> Fshapecoords
function(coords,base.0=1,base.1=2)
{
    (c2-c2[,base.0])/(c2[,base.1]-c2[,base.0])
}
```

C.7 Farea

This little routine implements the formula $\frac{1}{2}\sum_i^k (x_i y_{i+1} - x_{i+1} y_i)$ (where the subscript $k+1$ is interpreted as 1) for the area of a closed polygon on the points $(x_1, y_1), \ldots, (x_k, y_k)$ in that order. When the input is a list of complex numbers, this is typed for Splus or R as follows:

```
> Farea
function(coords)
{
    nland<-dim(coords)[[2]]
    0.5*apply(Re(coords)*Im(coords[,c(2:nland,1)])-
          Im(coords)*Re(coords[,c(2:nland,1)]),1,sum)
}
```

C.8 Ftpsoutlinelmks

This is the last of the basic algebraic spline manipulations that the main text introduces – it is delayed until Section 5.5, when the semilandmarks are introduced. The code is instructive enough to be worth including in this appendix. `Ftpsoutlinelmks` is called, as the name suggests, with three arguments: a configuration of left points `lpoints`, a commensurate configuration of right points `rpoints`, and a list `sliplist` of the entries of these lists that are *semilandmarks* allowed to slide. (The real landmarks are presumably the points that are not slipping.) Many versions of this routine add a step that projects the slid points `rpointsmin` down onto some sort of smoothed or splined version of the curve presumed to have been responsible for generating the list of right points; my version does not. Sliding directions are simply set by the chords under the semilandmarks. You can follow the algebra of Section 5.5.4 in the Splus code. This routine was used, along with a color randomization, to produce Figure 5.71.

```
> Ftpsoutlinelmks
function(lpoints,rpoints,sliplist)
{
    npt<-length(lpoints)
    nslip<-length(sliplist)
    linv<-solve(FLmatrix(lpoints))[1:npt,1:npt]
    xchord<-rpoints[c(2:npt,1)]
           -rpoints[c(npt,1:(npt-1))]
    biglinv<-rbind(cbind(linv,matrix
                  (0,nrow=npt,ncol=npt)),
          cbind(matrix(0,nrow=npt,ncol=npt),linv))
    bigz<-c(Re(rpoints),Im(rpoints))
    bigy<-matrix(0,ncol=nslip,nrow=2*npt)
    for (i in 1:nslip) {
    bigy[sliplist[i],i]<-Re(xchord[sliplist[i]])
```

```
    bigy[sliplist[i]+npt,i]<-Im(xchord[sliplist[i]])
    }
    tmin<-rep(0,npt)
    tmin[sliplist]<- -solve(t(bigy)%*%biglinv%*%bigy,
                      t(bigy)%*%biglinv%*%bigz)
    rpointsmin<-rpoints+tmin*xchord
    be1<-t(bigz)%*%biglinv%*%bigz
    be2<-t(c(Re(rpointsmin),Im(rpointsmin)))
      %*%biglinv%*%c(Re(rpointsmin),Im(rpointsmin))
    list(params=tmin,points=rpointsmin,be1=be1,
      be2=be2)
}
```

Variant version: `Ftpsoutline`, no sliplist – no actual landmark points, everything a semilandmark allowed to slide. This is the version used with the footprint data set.

C.9 Fpls2

This routine repackages the singular-value decomposition of the crosscovariance matrix of two blocks X, Y of data with the same count of rows (cases). It relies on the built-in Splus function `svd`, which returns a structure with elements `u`, `d`, and `v` in the eponymous slots.

```
> Fpls2
function(X,Y)
{
    s.v.d.<-svd(var(X,Y))
    xscores<-X%*%s.v.d.$u
    yscores<-Y%*%s.v.d.$v
    list(d=s.v.d.$d,xscores=xscores,yscores=yscores)
}
```

C.10 circle

One line:

```
circle<-complex(r=cos((0:100)*2*pi/100),
  i=sin((0:100)*2*pi/100))
```

This construction is easily modified to supply ellipses corresponding to covariance structures in two dimensions:

```
some.data.matrix<-matrix(some.data,N,2)
some.mean<-mean(complex(r=some.data.matrix[,1],
```

```
  i=some.data.matrix[,2]))
some.cov.structure<-princomp(some.data.matrix)
lines(some.mean+complex(
    r=some.cov.structure$loadings[1,1],
    i=some.cov.structure$loadings[2,1])
      *complex(r=some.cov.structure$sdev[1]*Re(circle),
      i=some.cov.structure$sdev[2]*Im(circle)))
```

as in Figure 3.44 of the text.

C.11 F.LLR2D

This is a dense routine for plotting the quadratic likelihood ratio surface corresponding to any pair of covariance structures on the same two variables. It is used for the book's Figure 3.44.

Appendix D Source Code for Select Figures

This final appendix includes my explicit source code for most of the empirical (i.e., data-based) figures in the book, plus a few others. These 124 files are intended as resources for readers who would like to emulate some of these analytic diagrams for their own data sets, but who do not wish to sort out the syntax of their graphical niceties and other details of presentation. Some of these fragments (you will consider these the more legible ones) simply pass elementary derived structures to standard plotting engines for lines or scatters. Others, which bundle the extraction of the essential geometrical parameters as prefatory command sequences, would surely be recoded as functions if they were to be used more than once, and still others render methodological points in the text or algrebraic properties of descriptors by diagrams of which I am particularly proud (e.g., Figure 2.24). The online resource is organized by chapter and subchapter as follows:

Chapter 2: Section 2.2, Figures 2.7 and 2.8; Section 2.3, Figures 2.9, 2.10, 2.12, and 2.24; Section 2.4, Figure 2.25, 2.28, and 2.31; Section 2.5, Figure 2.36; Section 2.6, Figures 2.37 through 2.40, 2.42 through 2.44, and 2.49.

Chapter 3: Section 3.1, Figures 3.1 through 3.9; Section 3.2, Figures 3.12 through 3.15; Section 3.3, Figures 3.16 through 3.19, 3.20c, 3.21, and 3.23 through 3.26; Section 3.4, Figures 3.27 through 3.32; Section 3.5, Figures 3.36 through 3.40; Section 3.6, Figures 3.43 through 3.45 and 3.47 through 3.51.

Chapter 4: Section 4.2, Figures 4.4, 4.7, 4.9, and 4.10; Section 4.3, Figures 4.16, 4.17, and 4.20 through 4.23.

Chapter 5: Section 5.1, Figures 5.3, 5.21, and 5.22; Section 5.2, Figures 5.29 through 5.36; Section 5.3, Figures 5.43 and 5.44; Section 5.4, Figures 5.47 through 5.60 (top); Section 5.5, Figures 5.64, 5.65, 5.67 through 5.71 and 5.73; Section 5.6, Figures 5.74 through 5.82, 5.84, 5.85, 5.87, 5.88, 5.92 through 5.95, and 5.97 through 5.99.