Functional principal components analysis of the bone shape data

We first construct matrices `landx` and `landy` containing the x and y coordinates of each of the 12 landmarks. These have been preprocessed so that the rows of landx corresponding to left femora are multiplied by –1, corresponding to reflection of the left bones. The next stage is to do a procrustes rotation to fit the various bones together as closely as possible. We start by transforming to zero mean on both coordinates:

```
> landx <- sweep( landx, 1, apply(landx, 1, mean))
> landy <- sweep( landy, 1, apply(landy, 1, mean))
```

We create a 68 x 12 x 2 array landa containing all the landmarks and then find the mean configuration corresponding to this matrix:

```
>  landa <- array(c(landx,landy), dim=c(68,12,2),
     dimnames=list(dimnames(landx)[[1]],(1:12),c("x","y")))

> meanconfig <- apply(landa, c(2,3), mean)
```

Now we carry out a procrustes rotation on each set of landmarks in turn, allowing for a magnification effect:

```
> landb <- landa
> for ( j in (1:68)) landb[j,,] <- procrustes(landa[j,,],
     meanconfig, T, T, magnify=T)$rmat
> meanconfig <- apply(landb, c(2,3), mean)
```

If we iterate this twice we achieve convergence, but note that the argument magnify should be set to F after the first or second iteration, otherwise the configuration simply shrinks at each iteration. Now we make landb back into a matrix and perform pca.

```
> landbb <- matrix( landb, nrow=68)

> landb.pca <- prcomp(landbb)
```

The function plotpc then plots the curves.

```
> plotpc <- function(pc = landb.pca, meancon = meanconfig)
{
      oldpars <- par()
      par(mfrow = c(2, 2), pty = "s", mar = rep(0.1,4), lwd = 2)
      for(comp in (1:4)) {
            load <- pc$rotation[, comp]
            load <- matrix(load, ncol = 2)
            sd <- pc$sdev[comp]
            zz <- meancon + 3 * sd * load
            zz <- rbind(zz, zz[1,  ])
            zzs1 <- spline(zz[, 1], n = 100, periodic = T)$y
            zzs2 <- spline(zz[, 2], n = 100, periodic = T)$y
            zz <- meancon - 3 * sd * load
            zz <- rbind(zz, zz[1,  ])
            zzs3 <- spline(zz[, 1], n = 100, periodic = T)$y
            zzs4 <- spline(zz[, 2], n = 100, periodic = T)$y
            plot(zzs1, zzs2, type = "l", xlim = range(c(zzs1, zzs3)
                  ), ylim = range(c(zzs2, zzs4)), axes = F, xlab
                   = "", ylab = "")
            box()
            lines(zzs3, zzs4, lty = 5, lwd = 2)
            text(150, 150, comp, cex = 2)
      }
      par(oldpars)
      invisible()
}
```

To get the rotated data, choose the matrix of principal component values and centre it at means:

```
> landb.rotx <- landb.pca$x
> landb.rotx <- sweep(landb.rotx, 2, apply(landb.rotx, 2, mean))
```

Now we can do a t test on each of the comparisons between eburnated and non-eburnated. Let `ebl` be a logical vector taking the value T for the eburnated bones. For the second component, for instance, we do

```
> t.test(landb.rotx[ebl, 2], landb.rotx[!ebl, 2])

      Standard Two-Sample t-Test
data:  landb.rotx[ebl, 2] and landb.rotx[!ebl, 2]
t = -3.0095, df = 66, p-value = 0.0037
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -23.707154  -4.796844
sample estimates:
 mean of x mean of y
 -10.89859  3.353412
```

To carry out a pca using a varimax rotation, we have to use a vector version of the quadrimax procedure. See the separate document on quadrimax rotation for a discussion of this, and for the function `vectorquadrimax`. If you want to do this for a one dimensional function, use the Splus function `orthomax` instead, selecting the `rmat` component of the result.

```
> varmxpc.fun <-
function(xdata = landb, ncomp = 5)
{
# first make x into a matrix if it isn't already one
      xd <- matrix(xdata, nrow = dim(xdata)[1])
# now do the pca of xd
      pc <- prcomp(xd)
# now do the varimax rotation on the first ncomp components
      rotation <- pc$rotation
      zorth <- vectorquadrimax(rotation[, 1:ncomp])
#  now replace the first ncomp columns of rotation
      rotation[, 1:ncomp] <- zorth
#  recalculate the scores
      x <- xd %*% rotation
      sdev <- sqrt(apply(x, 2, var))
      return(sdev, rotation, x)
}
```