Part VIII
Ordination

The package *oz*, used to draw a map of Australia, must be installed. In order to use the dataframe `diabetes` from the package *mclust*, that package must be installed. Or you can obtain the R image file from `"http://www.maths.anu.edu.au/~johnm/courses/dm/math3346/data/` Also required are the functions `read.dna()` and `dist.dna()` from the package *ape*; that package must be installed.

Where there is no "natural" measure of distance between points, and there are groups in the data that correspond to categorizations that will be of interest when the data are plotted, some form of discriminant analysis may be the preferred starting point for obtaining a low-dimensional representation. The low-dimensional representation that is likely to be superior to that obtained from ordination without regard to any such categorization.

Two of the possibilities that the ordination methods considered in this laboratory addresses are:

- Distances may be given, from which it is desired to recover a low-dimensional representation.
  - For example we may, as below, attempt to generate a map in which the distances on the map accurately reflect Australian road travel distances.
  - Or we may, based on genomic differences, derive genomic "distances " between, e.g., different insect species. The hope is that, with a judicious choice of the distance measure, the distances will be a monotone function of the time since the two species separated. We'd like to derive a 2 or 3-dimensional representation in which the distances accurately reflect the closeness of the evolutionary relationships.
- There may be no groupings of the data that are suitable for use in deriving a low-dimensional representation. Hence we calculate, using a metric that seems plausible, a matrix of distances between pairs of points, from which we in turn try to derive a low-dimensional representation.

1 Australian road distances

The distance matrix that will be used is in the matrix `audists`, in the image file **audists.Rdata**.

Here is how the data can be read in from the text file:

```
audists <- read.table("audists.txt", sep="\t")
audists[is.na(audists)] <- 0
## Also, we will later use the data frame aulatlong
aulatlong <- read.table("aulatlong.txt", row.names=1)[,2:1]
aulatlong[,2] <- -aulatlong[,2]
colnames(aulatlong) <- c("latitude", "longitude")
```

Consiser first the use of classical multi-dimensional scaling, as implemented in the function `cmdscale()`:

```
> library(DAAGxtras)
> aupoints <- cmdscale(audists)
> plot(aupoints)
> text(aupoints, labels = paste(rownames(aupoints)))
```

An alternative to `text(aupoints, labels=paste(rownames(aupoints)))`, allowing better placement of the labels, is `identify(aupoints, labels=rownames(aupoints))`. We can compare the distances in the 2-dimensional representation with the original road distances:

```
> origDists <- as.matrix(audists)
> audistfits <- as.matrix(dist(aupoints))
> misfit <- audistfits - origDists
> for (j in 1:9) for (i in (j + 1):10) {
```

```
+     lines(aupoints[c(i, j), 1], aupoints[c(i, j), 2], col = "gray")
+     midx <- mean(aupoints[c(i, j), 1])
+     midy <- mean(aupoints[c(i, j), 2])
+     text(midx, midy, paste(round(misfit[i, j])))
+ }
> colnames(misfit) <- abbreviate(colnames(misfit), 6)
> print(round(misfit))
```

	Adelad	Alice	Brisbn	Broome	Cairns	Canbrr	Darwin	Melbrn	Perth	Sydney
Adelaide	0	140	-792	-156	366	20	11	82	482	-273
Alice	140	0	-1085	-175	-41	76	-118	106	-26	-314
Brisbane	-792	-1085	0	198	319	-25	-233	-471	153	-56
Broome	-156	-175	198	0	527	-7	6	-65	990	70
Cairns	366	-41	319	527	0	277	-31	178	8	251
Canberra	20	76	-25	-7	277	0	-1	-241	372	-8
Darwin	11	-118	-233	6	-31	-1	0	-12	92	-58
Melbourne	82	106	-471	-65	178	-241	-12	0	301	-411
Perth	482	-26	153	990	8	372	92	301	0	271
Sydney	-273	-314	-56	70	251	-8	-58	-411	271	0

The graph is a tad crowded, and for detailed information it is necessary to examine the table.

It is interesting to overlay this "map" on a physical map of Australia.

```
> if (!exists("aulatlong")) load("aulatlong.RData")
> library(oz)
> oz()
> points(aulatlong, col = "red", pch = 16, cex = 1.5)
> comparePhysical <- function(lat = aulatlong$latitude, long = aulatlong$longitude,
+     x1 = aupoints[, 1], x2 = aupoints[, 2]) {
+     fitlat <- predict(lm(lat ~ x1 + x2))
+     fitlong <- predict(lm(long ~ x1 + x2))
+     x <- as.vector(rbind(lat, fitlat, rep(NA, 10)))
+     y <- as.vector(rbind(long, fitlong, rep(NA, 10)))
+     lines(x, y, col = 3, lwd = 2)
+ }
> comparePhysical()
```

An objection to `cmdscale()` is that it gives long distances the same weight as short distances. It is just as prepared to shift Canberra around relative to Melbourne and Sydney, as to move Perth. It makes more sense to give reduced weight to long distances, as is done by `sammon()` (*MASS*).

```
> library(MASS)
> aupoints.sam <- sammon(audists)

Initial stress        : 0.01573
stress after  10 iters: 0.00525, magic = 0.500
stress after  20 iters: 0.00525, magic = 0.500

> oz()
> points(aulatlong, col = "red", pch = 16, cex = 1.5)
> comparePhysical(x1 = aupoints.sam$points[, 1], x2 = aupoints.sam$points[,
+     2])
```

Notice how Brisbane, Sydney, Canberra and Melbourne now maintain their relative positions much better.

Now try full non-metric multi-dimensional scaling (MDS). THis preserves only, as far as possible, the relative distances. A starting configuration of points is required. This might come from the configuration used by `cmdscale()`. Here, however, we use the physical distances.

```
> oz()
> points(aulatlong, col = "red", pch = 16, cex = 1.5)
> aupoints.mds <- isoMDS(audists, as.matrix(aulatlong))

initial  value 11.875074
iter   5 value 5.677228
iter  10 value 4.010654
final  value 3.902515
converged

> comparePhysical(x1 = aupoints.mds$points[, 1], x2 = aupoints.mds$points[,
+     2])
```

Notice how the distance between Sydney and Canberra has been shrunk quite severely.

2 If distances must first be calculated ...

There are two functions that can be used to find distances – `dist()` that is in the base *statistics* package, and `daisy()` that is in the *cluster* package. The function `daisy()` is the more flexible. It has a parameter `stand` that can be used to ensure standardization when distancces are calculated, and allows columns that are factor or ordinal. Unless measurements are comparable (e.g., relative growth, as measured perhaps on a logarithmic scale, for different body measurements), then it is usually desirable to standardize before using ordination methods to examine the data.

```
> library(cluster)
> library(mclust)

use of mclust requires a license agreement
see http://www.stat.washington.edu/mclust/license.txt

> data(diabetes)
> diadist <- daisy(diabetes[, -1], stand = TRUE)
> plot(density(diadist, from = 0))
```

3 Genetic Distances

Here, matching genetic DNA or RNA or protein or other sequences are available from each of the different species. Distances are based on probabilistic genetic models that describe how gene sequences change over time. The package *ape* implements a number of alternative measures. For details see `help(dist.dna)`.

3.1 Hasegawa's selected primate sequences

The sequences were selected to have as little variation in rate, along the sequence, as possible. The sequences are available from:
`http://evolution.genetics.washington.edu/book/primates.dna`. They can be read into R as:

```
> library(ape)
> webpage <- "http://evolution.genetics.washington.edu/book/primates.dna"
> test <- try(readLines(webpage)[1])
> if (!inherits(test, "try-error")) {
+     primates.dna <- read.dna(con <- url(webpage))
+     close(con)
+     hasdata <- TRUE
+ } else hasdata <- FALSE
```

Now calculate distances, using Kimura's F84 model, thus

```
> if (hasdata) primates.dist <- dist.dna(primates.dna, model = "F84")
```

We now try for a two-dimensional representation, using `cmdscale()`.

```
> if (hasdata) {
+     primates.cmd <- cmdscale(primates.dist)
+     eqscplot(primates.cmd)
+     rtleft <- c(4, 2, 4, 2)[unclass(cut(primates.cmd[, 1], breaks = 4))]
+     text(primates.cmd[, 1], primates.cmd[, 2], row.names(primates.cmd),
+         pos = rtleft)
+ }
```

Now see how well the distances are reproduced:

```
> if (hasdata) {
+     d <- dist(primates.cmd)
+     sum((d - primates.dist)^2)/sum(primates.dist^2)
+ }
```

```
[1] 0.1977138
```

This is large enough (20%, which is a fraction of the total sum of squares) that it may be worth examining a 3-dimensional representation.

```
> if (hasdata) {
+     primates.cmd <- cmdscale(primates.dist, k = 3)
+     cloud(primates.cmd[, 3] ~ primates.cmd[, 1] * primates.cmd[,
+         2])
+     d <- dist(primates.cmd)
+     sum((d - primates.dist)^2)/sum(primates.dist^2)
+ }
```

```
[1] 0.1045168
```

Now repeat the above with `sammon()` and `mds()`.

```
> if (hasdata) {
+     primates.sam <- sammon(primates.dist, k = 3)
+     eqscplot(primates.sam$points)
+     rtleft <- c(4, 2, 4, 2)[unclass(cut(primates.sam$points[,
+         1], breaks = 4))]
+     text(primates.sam$points[, 1], primates.sam$points[, 2],
+         row.names(primates.sam$points), pos = rtleft)
+ }
```

```
Initial stress        : 0.11291
stress after  10 iters: 0.04061, magic = 0.461
stress after  20 iters: 0.03429, magic = 0.500
stress after  30 iters: 0.03413, magic = 0.500
stress after  40 iters: 0.03409, magic = 0.500
```

There is no harm in asking for three dimensions, even if only two of them will be plotted.

```
> if (hasdata) {
+     primates.mds <- isoMDS(primates.dist, primates.cmd, k = 3)
+     eqscplot(primates.mds$points)
```

```
+     rtleft <- c(4, 2, 4, 2)[unclass(cut(primates.mds$points[,
+         1], breaks = 4))]
+     text(primates.mds$points[, 1], primates.mds$points[, 2],
+         row.names(primates.mds$points), pos = rtleft)
+ }

initial  value 19.710924
iter   5 value 14.239565
iter  10 value 11.994621
iter  15 value 11.819528
iter  15 value 11.808785
iter  15 value 11.804569
final  value 11.804569
converged
```

.....
.....

The two remaining examples are optional extras.

4 *Distances between fly species

These data have been obtained from databases on the web. We extract them from an Excel **.csv** file (**dipt.csv**) and store the result in an object of class `"dist"`. The file **dipt.csv** is available from `http://www.maths.anu.edu.au/~johnm/datasets/ordination`.

Only below diagonal elements are stored, in column dominant order, in the distance object `dipdist` that is created below. For manipulating such an object as a matrix, should this be required, `as.matrix()` can be used to turn it into a square symmetric matrix. Specify, e.g., `dipdistM <- as.matrix(dipdist)`. For the clustering and ordination methods that are used here, storing it as a distance object is fine.

```
> webpage <- "http://www.maths.anu.edu.au/~johnm/datasets/ordination/dipt.csv"
> test <- try(readLines(webpage)[1])
> if (!inherits(test, "try-error")) {
+     diptera <- read.csv(webpage, comment = "", na.strings = c(NA,
+         ""))
+     dim(diptera)
+     species <- as.character(diptera[, 1])
+     table(substring(species, 1, 1))
+     species <- substring(species, 2)
+     genus <- as.character(diptera[, 2])
+     dipdist <- as.dist(diptera[1:110, 5:114])
+     attributes(dipdist)$Labels <- species
+     length(dipdist)
+     hasdata <- TRUE
+ } else hasdata <- FALSE
```

Two of the functions that will be used – `sammon()` and `isoMDS()` – require all distances to be positive. Each zero distance will be replaced by a small positive number.

```
> if (hasdata) dipdist[dipdist == 0] <- 0.5 * min(dipdist[dipdist >
+     0])
```

Now use (i) classical metric scaling; (ii) sammon scaling; (iii) isometric multi-dimensional scaling, with i as the starting configuration; (iv) isometric multi-dimensional scaling, with ii as the starting configuration.

```
> if (hasdata) {
+     dipt.cmd <- cmdscale(dipdist)
+     genus <- sapply(strsplit(rownames(dipt.cmd), split = "_",
+         fixed = TRUE), function(x) x[1])
+     nam <- names(sort(table(genus), decreasing = TRUE))
+     genus <- factor(genus, levels = nam)
+     plot(dipt.cmd, col = unclass(genus), pch = 16)
+     dipt.sam <- sammon(dipdist)
+     plot(dipt.sam$points, col = unclass(genus), pch = 16)
+     dipt.mds <- isoMDS(dipdist, dipt.cmd)
+     plot(dipt.mds$points, col = unclass(genus), pch = 16)
+     dipt.mds2 <- isoMDS(dipdist, dipt.sam$points)
+     plot(dipt.mds2$points, col = unclass(genus), pch = 16)
+ }
```

5 *Rock Art

Here, we use data that were collected by Meredith Wilson for her PhD thesis. The 614 features were all binary – the presence or absence of specific motifs in each of 103 Pacific sites. It is necessary to omit 5 of the 103 sites because they have no motifs in common with any of the other sites. Data are in the *DAAGxtras* package.

The binary measure of distance was used – the number of locations in which only one of the sites had the marking, as a proportion of the sites where one or both had the marking. Here then is the calculation of distances:

```
> library(DAAGxtras)
> pacific.dist <- dist(x = as.matrix(rockArt[-c(47, 54, 60, 63,
+     92), 28:641]), method = "binary")
> sum(pacific.dist == 1)/length(pacific.dist)

[1] 0.6311803

> plot(density(pacific.dist, to = 1))
> symmat <- as.matrix(pacific.dist)
> table(apply(symmat, 2, function(x) sum(x == 1)))

13 21 27 28 29 32 33 35 36 38 40 41 42 43 44 45 46 47 48 49 51 52 53 54 55 56
 1  1  1  1  2  1  2  1  2  2  1  2  4  3  1  3  1  2  1  1  2  2  3  2  2  2
57 58 61 62 64 65 66 67 68 69 70 71 73 75 76 77 79 81 83 84 85 90 91 92 93 94
 1  3  3  1  2  1  1  1  3  3  1  1  4  1  2  1  1  1  2  1  1  3  1  1  3  1
95 96 97
 1  3  4
```

It turns out that 63% of the distances were 1. This has interesting consequences, for the plots we now do.

```
> pacific.cmd <- cmdscale(pacific.dist)
> eqscplot(pacific.cmd)
```