36-350: Data Mining

Lab 6
Date: October 4, 2002 **Due: end of lab**

---

1 Introduction

This lab teaches you the steps involved in visualizing subgroups of a dataset.
There are 6 questions. For each one, submit your commands and a response from R demonstrating that they work. (Only hand in commands relevant to the question.) To submit a plot, click on the plot window and select

```
File -> Save as -> Postscript...
```

This saves the plot to a file which can be printed, incorporated into a Word document, or mailed to us as an attachment.

2 Starting R

Start R as in lab 1. On the class web page, go to "computer labs" and download the files for lab 6 into your work folder. Read the special functions into your running R application via the commands

```
source("lab4.r")
source("lab5.r")
source("lab6.r")
```

If this fails, check that the files were downloaded correctly.

3 The data

The dataset used in this lab is the 50 states, as in lab 5. Load this data via

```
load("States.rda")
```

This defines a matrix called `states` and a factor called `state.region`. As in lab 5, the variables must first be standardized:

```
i <- c("Illiteracy","Density")
states[,i] <- log(states[,i])
sx <- scale(states)
```

Now `sx` has the standardized data.

4 Clustering

In this lab, we want to know how the geographic division of states, given by `state.region`, compares to the demographic division given by clustering `states`. There are four regions in `state.region`, so to make the comparison you want to divide `sx` into four clusters.
To cluster this data, recall the commands from lab 4:

```
d <- dist(sx)^2
hc <- hclust(d,method="ward")
plot(hc)
plot.hclust.trace(hc)
f <- factor(cutree(hc,(number of clusters)))
sum.of.squares(sx,f)
```

The result, `f`, is a factor of cluster numbers.
The first step in comparing `f` to `state.region` is to make a contingency table. The function `table` constructs a table from two factors:

```
table(cluster=f,region=state.region)
```

> **Question 1:** For each of the clusters, which region is most common in that cluster? Use this to give a region name to each cluster. The command to change the names in a factor is
>
> ```
> attr(f,"levels") <- c("name of cluster 1","name of cluster 2",etc)
> ```
>
> Try not to have the same name for two different clusters.
>
> **Question 2:** How does the sum of squares for Ward's method compare to the sum of squares from dividing the states according to `state.region`?

5 PCA projection

Recall the commands for PCA projection:

```
w <- pca(sx,(final number of dimensions))
px <- project(sx,w)
text.plot(px[,1],px[,2],rownames(px),cex=0.8)
plot.axes(w,col=3)
```

Now we want to color the data according to its cluster. This is done with the command `color.plot`. It is used similar to `plot`:

```
color.plot(x,y,f)
```

Here `x`, `y`, and `f` are vectors of the same length. `x` and `y` are numeric vectors, describing position, and `f` is a factor, describing subgroups to color. `main` is the label that appears above the color key. Instead of dots, `color.plot` can also show labels, just like `text.plot`. The difference from above is that you provide `labels` and a character expansion factor `cex`:

```
color.plot(x,y,f,labels,cex=0.8)
```

(Recall the function `rownames`.)

> **Question 3:** Make two PCA plots: one with the projected data colored according the clustering and another with the projected data colored according to `state.region`. Show the state names in each case. Turn in the plots and keep a copy for the homework. (Be careful about printing: color plots can be difficult to read when printed in black and white. You won't get credit for an unreadable plot.)

6 Discriminative projection

To examine the shape of a cluster, it helps to make a different projection, one that maximally separates the cluster from the rest of the data. The function for this is `projection`, which is used as follows:

```
w <- projection(sx,f,k=2,type="m")
```

Here `f` is a factor describing the groups you want to separate. `f` can contain many groups or only two groups. `k` is the number of dimensions to project down to, like `pca`. `type` describes the distribution assumption to be used. Here we will assume the clusters are normal with equal variance, so only the means need to be separated. This is type "m". The result is a projection matrix, just like from `pca`. To separate one cluster from everything else, you need to create a factor which represents this two-group division. This is done by the function `factor.dichotomy`:

```
fd <- factor.dichotomy(f,"West")
```

The result is a factor which groups the states into `West` and `NOT West`, as defined by `f`. `fd` can be given to `projection`.

> **Question 4:** Make two discriminative projection plots: one which separates `South` from the rest of the data and one which separates `Northeast`. The projections should be different. You can color the states according to `f` or `fd`. `fd` might be better if you are going to print out the plot.

7 Parallel-coordinate plots

A parallel-coordinate plot is not useful on the entire dataset, so first you take cluster prototypes:

```
xp <- prototypes(sx,f)
```

The command for a parallel-coordinate plot is `parallel.plot`, and it has a variety of options for how to shift and scale the variables on the plot:

```
parallel.plot(xp)
parallel.plot(xp,xscale="equal")
parallel.plot(xp,xscale="none",yscale="range")
```

The first version tries to make the curves as straight as possible. The second version is the same but forces the variables to be spaced equally, for easier reading. This is recommended. The third version demonstrates a "naive" parallel-coordinate plot, where the variables are shown in their default order and scaled based on their range (they are never reversed). The plot can be switched to black and white by adding the argument `color.palette=1`.
If the labels in the above plots are too big, try typing this:

```
par(cex=0.8,cex.axis=0.8)
```

This will cause R to plot smaller text from now on.

> **Question 5:** Turn in (and save) a parallel-coordinate plot of the cluster prototypes from Ward's method.

Instead of just prototypes, you can show individual states on the plot. The following commands will create a new clustering in which Texas is taken out of its previous cluster and put by itself (it is "isolated"):

```
names(f) <- rownames(sx)
f2 <- factor.isolate(f, "Texas")
```

If you compute the prototypes of this clustering and make a parallel-coordinate plot, it will show how Texas compares to the four clusters without Texas.

> **Question 6:** Make two parallel-coordinate plots: one where Texas is isolated and another where New Mexico is isolated. Note that the variable ordering and scaling may change slightly between the plots. Turn them in and save for the homework.