Exploratory Projection Pursuit

We are now going to look at an exploratory tool called projection pursuit (Jerome Friedman, *PROJECTION PURSUIT METHODS FOR DATA ANALYSIS*, June 1980, SLAC PUB-2768)

Read in required files:

```r
drive <- "D:"
code.dir <- paste(drive, "DATA/Data Mining R-Code", sep="/")
data.dir <- paste(drive, "DATA/Data Mining Data", sep="/")
source(paste(code.dir, "3DRotations.r", sep="/"))
source(paste(code.dir, "EllipseCloud.r", sep="/"))
source(paste(code.dir, "BorderHist.r", sep="/"))
source(paste(code.dir, "CreateGaussians.r", sep="/"))
source(paste(code.dir, "ReadFleas.r", sep="/"))
```

and set up a couple of required functions.

The first one will *centre the data and ensure unit variance* (i.e. sphere the data).

```r
# Centre and sphere data
Sphere.Data <- function(data) {
  data <- as.matrix(data)
  data <- t(t(data) - apply(data, 2, mean))
  data.svd <- svd(var(data))
  sphere.mat <- t(data.svd$v) %*% (t(data.svd$u) * (1/sqrt(data.svd$d))))
  return(data %*% sphere.mat)
}
```

(Recall that we had an earlier function that does a similar thing.)

```r
# Standardize data
f.data.std <- function(data) {
  data <- as.matrix(data)
  bar <- apply(data, 2, mean)
  s <- apply(data, 2, sd)
  t((t(data) - bar)/s)
}```
It might be worthwhile to look at *how the two differ* by considering an example. The following reads in an elliptic cloud:

```r
data.prp <- read.table(file=paste(data.dir, "PPellipse.tab", sep="/"))
```

and plots it

```r
plot(data.prp, xlim=c(-30,30), ylim=c(-30,30))
```

![Figure 1. Elliptic cloud](image1.png)

![Figure 2. Elliptic cloud with ID and extreme points](image2.png)

It is possible to determine which of the data points are “extreme” by `identify(data.prp)`

Go to the plot. Your mouse will now display a “+” sign. Click on one of the extreme points and the observation corresponding to it will show on the graph. Once you are finished selecting the extreme points, to exit the `identify` function, right click in the Graphics window and select “Stop” from the menu that appears. Now “colour” these newly found points (Figure 2):

```r
points(data.prp[220,], pch=8, col="red")
points(data.prp[189,], pch=7, col="red")
points(data.prp[216,], pch=8, col="blue")
points(data.prp[100,], pch=7, col="blue")
```

We can now standardize (as required):

```r
data.prp.std <- f.data.std(data.prp)
apply(data.prp.std, 2, mean)
  X1   X2
5.392388e-17 2.597714e-16
apply(data.prp.std, 2, sd)
  X1  X2
1   1
```
and sphere (as required):

data.prp.sph <- Sphere.Data(data.prp)
colnames(data.prp.sph) <- list("X1", "X2")
apply(data.prp.sph, 2, mean)

X1  X2
1.125332e-16 -2.181661e-16

apply(data.prp.sph, 2, sd)

X1  X2
1 1

Now plot them and show what happens to the extreme values:

plot(data.prp.std, asp=1, main="Standardized", xlim=c(-4, 4), ylim=c(-4, 4))
points(matrix(data.prp.std[220,], ncol=2), pch=8, col="red")
points(matrix(data.prp.std[189,], ncol=2), pch=7, col="red")
points(matrix(data.prp.std[216,], ncol=2), pch=8, col="blue")
points(matrix(data.prp.std[100,], ncol=2), pch=7, col="blue")
x11()
plot(data.prp.sph, asp=1, main="Sphered", xlim=c(-4, 4), ylim=c(-4, 4))
points(matrix(data.prp.sph[220,], ncol=2), pch=8, col="red")
points(matrix(data.prp.sph[189,], ncol=2), pch=7, col="red")
points(matrix(data.prp.sph[216,], ncol=2), pch=8, col="blue")
points(matrix(data.prp.sph[100,], ncol=2), pch=7, col="blue")

![Figure 3.](image1.png)

![Figure 4.](image2.png)

We can see that the standardized “cloud” has retained its elliptic shape and orientation while the sphered “cloud” is more circular and that lines joining the blue points and joining the red would appear to cross at right angles. The re-orientation comes from the use of the eigenvectors which, as we saw in the PCA, identifies the directions of decreasing variance and re-orientes them to be orthogonal.
We are going to look at a second new function gives an **index** that allows us to determine if the data has most of its information on either side of the centre (see later):

```r
# Holes Index
Holes.Index <- function(data) {
  1 - mean(exp(-data^2))
}
```

We will create two elliptic clouds by use of the functions in `3DRotations.r` and `CreateGaussians.r`:

```r
# 2D rotation
R.2D <- function(data, Ang) {
  T <- diag(rep(1, 2))      # 2 x 2 identity matrix
  c <- cos(Ang)
  s <- sin(Ang)
  T[1, 1] <- c               #
  T[2, 2] <- c               # cos  sin
  T[2, 1] <- -s              # -sin  cos
  T[1, 2] <- s               #
  data %*% T
}

# Create a Gaussian cloud mean (x,y) variance (varx,vary)
# which is rotated by the angle rot
Make.Gaussian <- function(n, x, y, varx=1, vary=1, rot) {
  X.1 <- rnorm(n, x, varx)
  Y.1 <- rnorm(n, y, vary)
  R.2D(cbind(X.1, Y.1), rot)
}
```

The first cloud has 500 points - mean (0, 0), variance (5, 3) - and is rotated by $\pi/4$, then centred at (5,15):

```r
sz.1 <- 500
data.1 <- Make.Gaussian(sz.1, 0, 0, varx=5, vary=3, pi/4)
data.1 <- t(t(data.1) + c(5,15))
```

The second cloud has 500 points - mean (0, 0), variance (5, 3) - and is rotated by $\pi/4$, then centred at (15,5):

```r
sz.2 <- 500
data.2 <- Make.Gaussian(sz.2, 0, 0, varx=5, vary=3, pi/4)
data.2 <- t(t(data.2) + c(15,5))
```
Combine the two clouds (\texttt{rbind} makes a 1000 point data set), and plot:

\begin{verbatim}
G.1.data <- rbind(data.1, data.2)
plot(G.1.data, asp=1, main="Two elliptic clouds")
\end{verbatim}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{two_elliptic_clouds.png}
\caption{Two elliptic clouds}
\end{figure}

Viewing then in the 2-d plane is like looking at the clouds from above. Suppose that we can only see them as \textit{projections on the x and y-axes}. Our way of viewing such data is as a \textit{histogram}. The following function will display the data and histograms together.

\begin{verbatim}
border.hist <- function(x, y) {
  # Create a graph with a histogram of the data on the border.
  oldpar <- par(no.readonly = TRUE) # Read the current state
  xhist <- hist(x, plot=FALSE, breaks = 20) # Create histograms
  yhist <- hist(y, plot=FALSE, breaks = 20)
  top <- max(c(xhist$counts, yhist$counts)) # Get max count
  xrange <- range(x) # min, max
  yrange <- range(y)
  # Create a plot canvas that has 2 rows and 2 columns
  # on which the c(0,2,3,1) gives the order of plotting
  # 0 = no plot, and c(1,3), c(1,3) give the row and col
  # size ratios.
  nf <- layout(matrix(c(0,2,3,1),2,2,byrow=TRUE), c(1,3), c(1,3), TRUE)
  layout.show(nf)
  # Set the margin sizes (bottom, left, top, right)
  par(mar=c(3,3,1,1))
  plot(x, y, xlim=xrange, ylim=yrange, xlab="", ylab="")
  par(mar=c(0,3,1,1))
  barplot(xhist$counts, axes=FALSE, ylim=c(0, top), space=0,
          col=heat.colors(length(xhist$breaks)))
  par(mar=c(3,0,1,1))
  barplot(yhist$counts, axes=FALSE, xlim=c(top, 0), space=0, horiz=TRUE,
          col=heat.colors(length(yhist$breaks)))
}
\end{verbatim}
par(oldpar)  # reset to default
}
border.hist(G.1.data[,1], G.1.data[,2])

Figure 6. Border Histogram for two unsphered clouds

Figure 7. Border Histogram for two sphered clouds

Figure 6 shows the border histograms for unsphered clouds; Figure 7 shows the same thing for the sphered data. Note that the number of bins is 15 rather than the 20 requested for the number of breaks in the call to `hist`. That number can be over-ridden by the function if it is not optimal.

G.1.data.sph <- Sphere.Data(G.1.data)
cor(G.1.data.sph)
   [,1]      [,2]
[1,] 1.000000e+00 -2.942152e-16
[2,] -2.942152e-16  1.000000e+00
cov(G.1.data.sph)  # Same result

border.hist(G.1.data.sph[,1], G.1.data.sph[,2])
We now wish to rotate our viewpoint (i.e. project the data onto different axes). For this purpose we could put the centre of the data at \((0,0)\) by subtracting the mean. This would not have any effect on the nature of the distribution of data.

We will also “sphere” the data in order to be able to quantify the ”appearance” of the histograms. The process of sphering is basically a PCA with scaling. **NOTE: This will change the spatial nature of the data.** The problem is that it may distort the characteristics that we wish to discover (as shown below).

```r
# Two small clouds
# Cloud 3
sz.3 <- 10
data.3 <- Make.Gaussian(sz.3, -8, 0, varx=3, vary=1, 0)

# Two small clouds
# Cloud 4
sz.4 <- 10
data.4 <- Make.Gaussian(sz.4, 8, 0, varx=3, vary=1, 0)
G.2.data <- rbind(data.3, data.4)
plot(G.2.data, xlim=c(-15,15), asp=1, main="Two small elliptic clouds")

plot(Sphere.Data(G.2.data), xlim=c(-15,15), asp=1, main="Two small elliptic clouds after sphering")
```

**Figure 8.** Two small clusters  
**Figure 9.** Two small clusters after sphering

We see that the clusters which seemed to be evident in Figure 3 have been pushed together in Figure 4. While this makes it difficult for us to visualize structure, it does not affect the Gaussian/non-Gaussian nature of the data so an index which looks for non-normality is not affected.
With this in mind consider the following:

G.1.data.sph <- Sphere.Data(G.1.data)
print(cor(G.1.data.sph))
print(cov(G.1.data.sph))

G.1.data.sph <- Sphere.Data(G.1.data)
print(cor(G.1.data.sph))
print(cov(G.1.data.sph))

border.hist(G.1.data.sph[,1], G.1.data.sph[,2])

Figure 10. Border Histogram for two clouds (sphered)

We notice that the data has what appears to be a normal distribution when projected on both the x and the y directions. We can see no structure in this data from this view.

Now suppose we look at the **projections on a sequence of lines set at angles to the x-axis**: 

oldpar <- par(mfrow = c(3,3))
par(mar=c(2,2,2,2)+0.1)
for (i in 0:8) {
  Ax <- cbind(c(cos(i*pi/6), sin(i*pi/6))) # Set up a rotation
  G.1.proj <- G.1.data.sph%*%Ax # Rotate the data
  h.i <- Holes.Index(G.1.proj)
  hist(G.1.proj, 50, main=paste(i,"*pi/6", " - Holes Index = ",floor(h.i*1000)/1000, sep=""));
}
Now we see that we can determine **structure** (a distinct gap) in the projections.

*Can we systematically search for such structure?* (It might be noted that there are different types of structure that we can look for but they all have a common characteristic - they are all **non-Gaussian**.)

In a situation such as this, we might use a ‘**Holes**’ index (i.e. look for a space in the middle). The Holes index simply takes $e^{-x^2}$ for every projected point, finds the average and subtracts from 1. Because the data has been centred, a Gaussian will have a small index, while a bimodal structure will have a larger one. If all the data is at the edges, the index will be even smaller. In another situation we might want to discover if the data is pushed to the middle or has a thick tail.

The **pursuit** part of the name comes from the fact that a maximization routine can be used so that, from any random projection, a **local maximum** for an appropriate index can be found. As usual, there is no assurance that the local maximum is the best one so **several restarts are needed**. The process here becomes similar to the Grand Tour in Ggobi. In the Grand Tour the direction vectors for the projection planes are selected randomly, while in projection pursuit an optimization routine is used to select the next direction vector for the projection.
Here we will look at flea beetle data in Ggobi; project the data onto different axes:

```r
library(rggobi)
g <- ggobi(d.flea)
```

Because *Ggobi uses indices that require sphering for the projection pursuit*, we select [Sphering] from the [Tools] menu.

![Ggobi sphering](image)

**Figure 12.** Ggobi sphering

This brings up a dialog box to enable us to sphere our data. In this, we select all the variables, then click [Update scree plot] (Figure 14) In Figure 15, we have clicked [Apply sphering, add PCs to data] to do the sphering.
**Figure 13.** Setup for sphering

**Figure 14.** PCA

**Figure 15.** Data sphered
We close the dialog box and see that the console has 6 new variables. We select a [2D Tour] from the [View] menu (Figure 16) with the original variables.

Figure 16. Select 2D Tour

Figure 17. Original variables selected

We now select the PC variables and deselect the original ones.

Figure 18. PC variables selected

Figure 19. Startup of tour on PC
To select/deselect variables to be included, we click on the X beside the name. In Figure 18, the new variables are selected. Figure 19 shows the data as it might appear on start-up and indicates that variable 7-12 (sphered) are used. In order to show the numbers rather than the name, on the scatterplot window, deselect [Show Axes Labels] from the [Tour2D] menu.

To start the pursuit, in the GGobi window, select [Projection pursuit]. This opens another window (Figure 20) which will display the selected index (numerically and graphically), allow us to change the index, and allow a choice of the tour proceeding as a random tour or of optimizing the index. Figure 21 shows what happens if we select optimize; here, Ggobi tries to optimize the Hole’s index.

**Figure 20.** Pursuit index window

**Figure 21.** Pursuit index at optimum

**Figure 22.** Clouds at optimum Holes Index

Figure 22 shows the three clusters once the index indicates an optimum value.