Multivariate Data

Jonathan M. Lees
University of North Carolina, Chapel Hill
Department of Geological Sciences
CB #3315, Mitchell Hall
Chapel Hill, NC 27599-3315
email: jonathan.lees@unc.edu
ph: (919) 962-0695

November 5, 2008

1 Discriminant Analysis

Read in the data from the disk provided by Davis. The relates grain size versus sorting order. A plot of the original SANDS data follows as Figure 1.

```
> sands = scan(file = "/home/lees/Class/Data_Analysis/dos_files/SANDS.TXT",
+     skip = 1, list(cat = "", gs = 0, so = 0))
```

Begin by calculating the required matrix by removing the means. There are 2 categories in the file, labeled, A and B. Bind the data together to form a matrix,

```
> d = c(mean(sands$gs[sands$cat == "A"]) - mean(sands$gs[sands$cat ==
+     "B"]), mean(sands$so[sands$cat == "A"]) - mean(sands$so[sands$cat ==
+     "B"]))
> na = length(sands$gs[sands$cat == "A"])
> nb = length(sands$gs[sands$cat == "B"])
> Va = cbind(sands$gs[sands$cat == "A"] - mean(sands$gs[sands$cat ==
+     "A"]), sands$so[sands$cat == "A"] - mean(sands$so[sands$cat ==
+     "A"]))
```

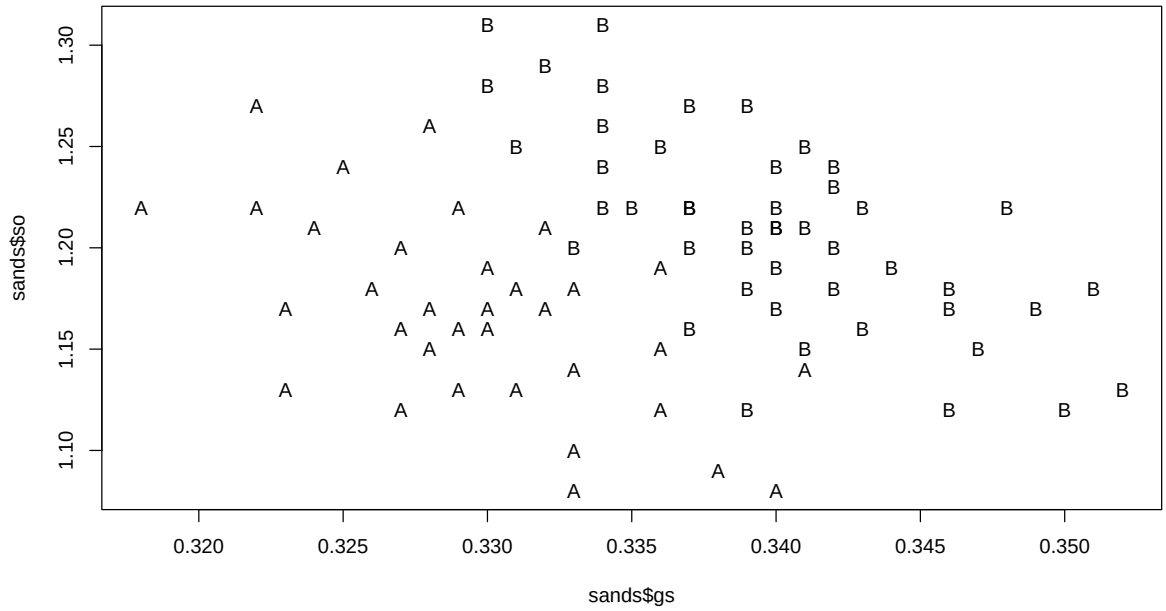

Figure 1: SAND

The pooled estimates are attained by adding the Va and Vb matrices.

```
> SPa = t(Va) %*% Va
> Vb = cbind(sands$gs[sands$cat == "B"] - mean(sands$gs[sands$cat ==
+     "B"]), sands$so[sands$cat == "B"] - mean(sands$so[sands$cat ==
+     "B"]))
> SPb = t(Vb) %*% Vb
> SPool = (SPa + SPb)/(na + nb - 2)
```

Use function "solve" as a standard way of solving for the inverse without resorting to library(MASS) (old way used a different function). The lambdas mentioned in the book are derived by multiplying the inverse of the matrix with the difference vector, d.

```
> invS = solve(SPool)
> lamb = invS %*% d
```

The solution can be represented as a line through the data that best separates the data into groups. If we project the data onto this line we can form a discriminantn function. The slope of the line is a ratio of eigenvalues λ_1, λ_2.

```
> slope = lamb[1]/lamb[2]
> xj = c((mean(sands$gs[sands$cat == "A"]) + mean(sands$gs[sands$cat ==
+     "B"]))/2, (mean(sands$so[sands$cat == "A"]) + mean(sands$so[sands$cat ==
+     "B"]))/2)
```

```
> postscript(file = "SANDS_2.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(sands$gs, sands$so, type = "p", pch = sands$cat)
> points(xj[1], xj[2], col = 2, pch = 6)
> points(mean(sands$gs[sands$cat == "A"]), mean(sands$so[sands$cat ==
+     "A"]), col = 4, pch = 6)
> points(mean(sands$gs[sands$cat == "B"]), mean(sands$so[sands$cat ==
+     "B"]), col = 4, pch = 6)
> cept = xj[2] - xj[1] * slope
> abline(cept, slope)
> dev.off()
```

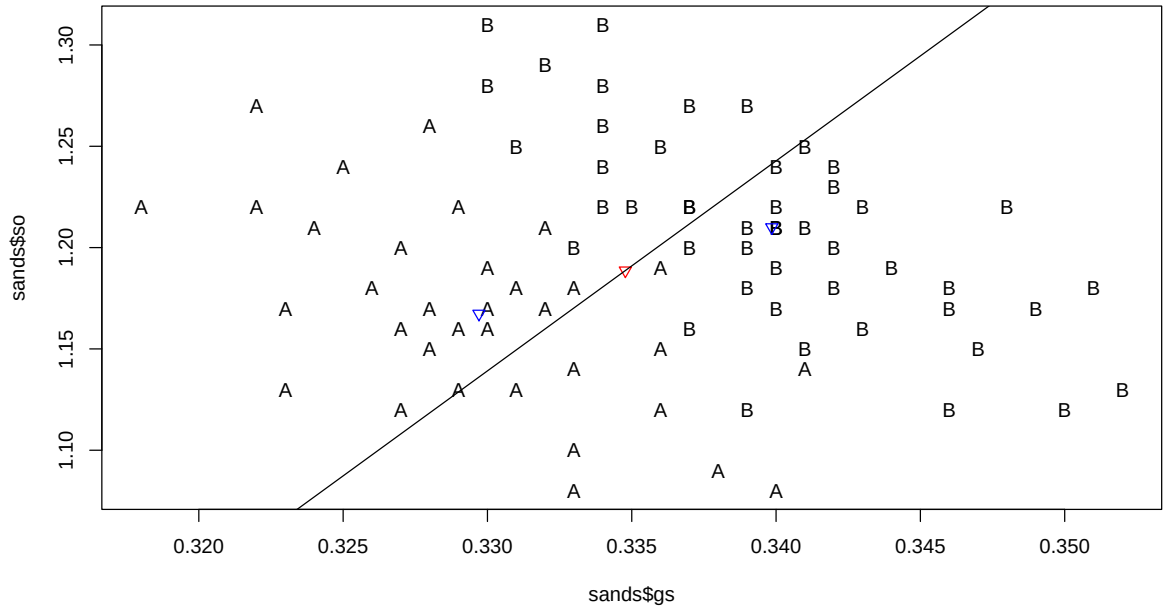

Figure 2: Discriminant Ananlysis

Next we consider the discriminant function and project the data so that it can be viewed as a 1 dimensional problem of discrimination. Setting up the projections of the discriminant scores:

```
> R0 = lamb[1] * xj[1] + lamb[2] * xj[2]
> RA = lamb[1] * mean(sands$gs[sands$cat == "A"]) + lamb[2] * mean(sands$so[sands$cat =
+     "A"])
> RB = lamb[1] * mean(sands$gs[sands$cat == "B"]) + lamb[2] * mean(sands$so[sands$cat =
+     "B"])
> Ri = t(lamb) %*% rbind(sands$gs, sands$so)
> Ri = as.vector(Ri)
> Y = abs(rnorm(length(Ri)))
> col = rep(2, length(sands$cat))
> col[sands$cat == "A"] = 4
```

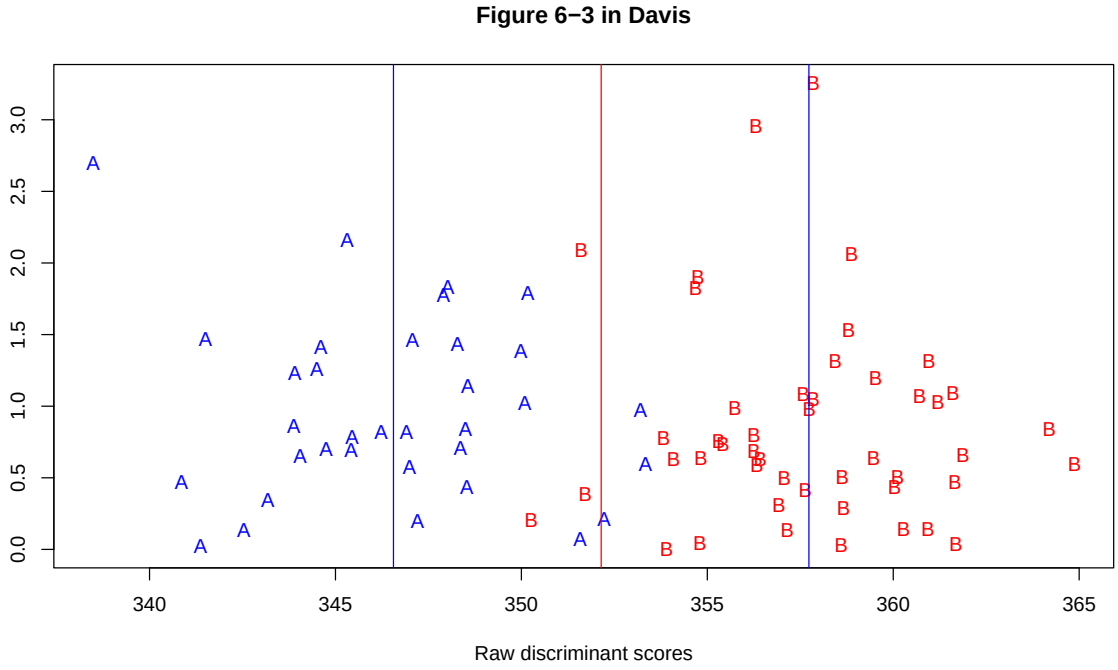

Figure 3: Projected Discriminant Function

All this can be achieved more elegantly throught PCA, or Prinicipal Component Analysis, in the next section.

2 PCA analysis

PCA is principal Component analysis. It mainly involves a rotation of the axes to a new basis that is a combination of the original data axes. To begin we will read in data from the Davis book and plot these to inspect the original data. Next the data will be normalized and centered, and PCA

will find a new set of coordinates to examine structure in the data. Read and plot the data from Table 6.12 in Davis.

```
> t16 = scan(file = "/home/lees/Class/Data_Analysis/dos_files/TABLE612.TXT",
+     skip = 1, list(x1 = 0, x2 = 0, r = 0, ran = 0))
> postscript(file = "TAB612.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(t16$x1, t16$x2, type = "p")
> grid()
> dev.off()
```

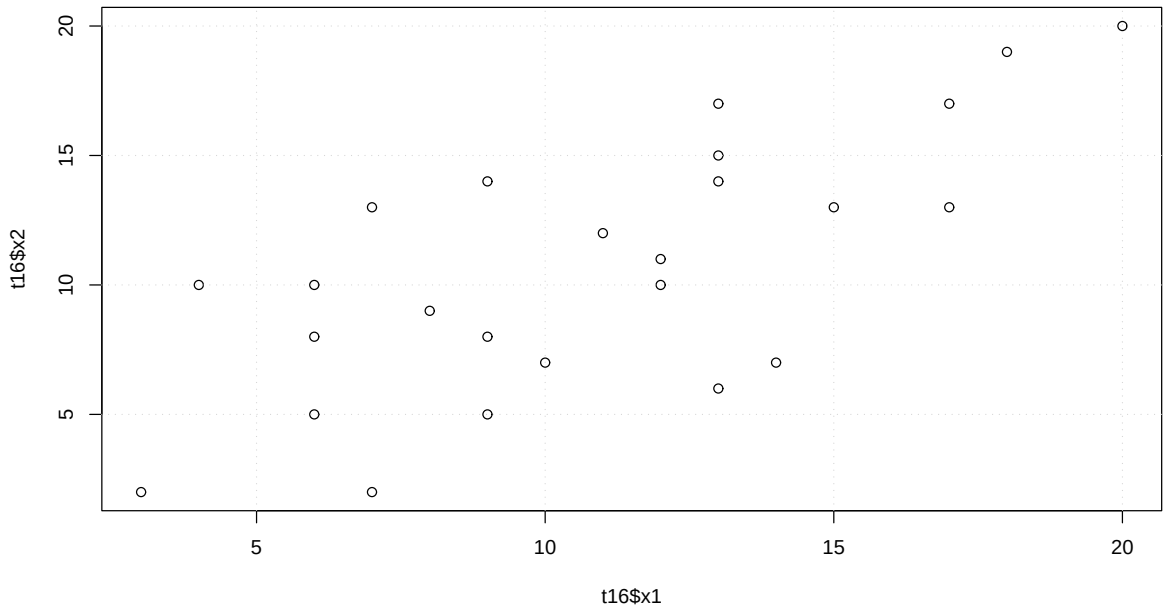

Figure 4: TAB612

Formulate the variance-covariance matrix and calculate the eigenvalues.

```
> v = cbind(t16$x1, t16$x2)
> sv = var(v)
> esv = eigen(sv)
> sr = v %*% esv$vectors
```

```
> postscript(file = "FIG6-16.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(c(-40, 40), c(-40, 40), asp = TRUE, type = "n")
> abline(h = 0)
> abline(v = 0)
> arrows(0, 0, sv[1, 1], sv[2, 1], col = 4, lty = 2)
> arrows(0, 0, -sv[1, 1], -sv[2, 1], col = 4, lty = 2)
> arrows(0, 0, sv[1, 2], sv[2, 2], col = 2, lty = 2)
> arrows(0, 0, -sv[1, 2], -sv[2, 2], col = 2, lty = 2)
> arrows(0, 0, esv$values[1] * esv$vectors[1, 1], esv$values[1] *
+     esv$vectors[2, 1], col = 4, lty = 1)
> arrows(0, 0, -esv$values[1] * esv$vectors[1, 1], -esv$values[1] *
+     esv$vectors[2, 1], col = 4, lty = 1)
> arrows(0, 0, esv$values[2] * esv$vectors[1, 2], esv$values[2] *
+     esv$vectors[2, 2], col = 2, lty = 1)
> arrows(0, 0, -esv$values[2] * esv$vectors[1, 2], -esv$values[2] *
+     esv$vectors[2, 2], col = 2, lty = 1)
```

Here create and draw the rotated ellipse:

```
> phi = -180 * atan2(esv$vectors[2, 1], esv$vectors[1, 1])/pi
> theta = seq(0, 360, by = 5) * pi/180
> cosp = cos(phi * pi/180)
> sinp = sin(phi * pi/180)
> a = esv$values[1]
> b = esv$values[2]
> r = matrix(c(cosp, sinp, -sinp, cosp), ncol = 2)
> m = matrix(rep(0, 2 * length(theta)), ncol = 2)
> m[, 1] = a * cos(theta)
> m[, 2] = b * sin(theta)
> nm = m %*% r
> lines(nm[, 1], nm[, 2], col = 3)
> title("Figure Davis 6-16, page 511")
> dev.off()
```

Since the orientation of the eigenvectors is arbitrary and depends on the computer program for a convention, the book uses a different direction than R. To get the figure in the book on page 513 (fig 6-17) flip the sign of the vectors,

```
> postscript(file = "FIG6-16_2.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
```

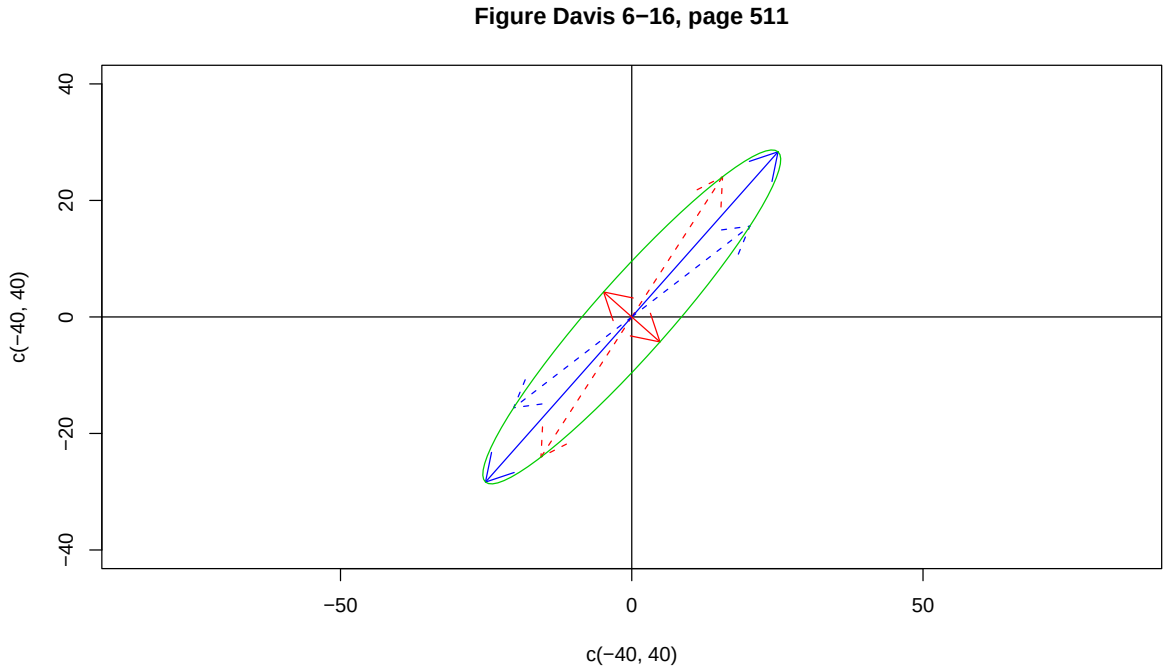

Figure 5: Figure Davis 6-16, page 511

```
> plot(sr[, 1], -sr[, 2])
> title("Davis  513 (fig 6-17)")
> dev.off()
```

2.1 Example: BARATARA.TXT

The file BARATARA.TXT consists of fifty Grain-Size Analyses of Bottom Sediment Collected in Barataria Bay, Louisiana; Samples are Classified into Five Types:

1. beach and foreshore sands;
2. silty channel sands;
3. silty channel margin sands;
4. organic bottom silts;
5. organic muds from lees of islands.

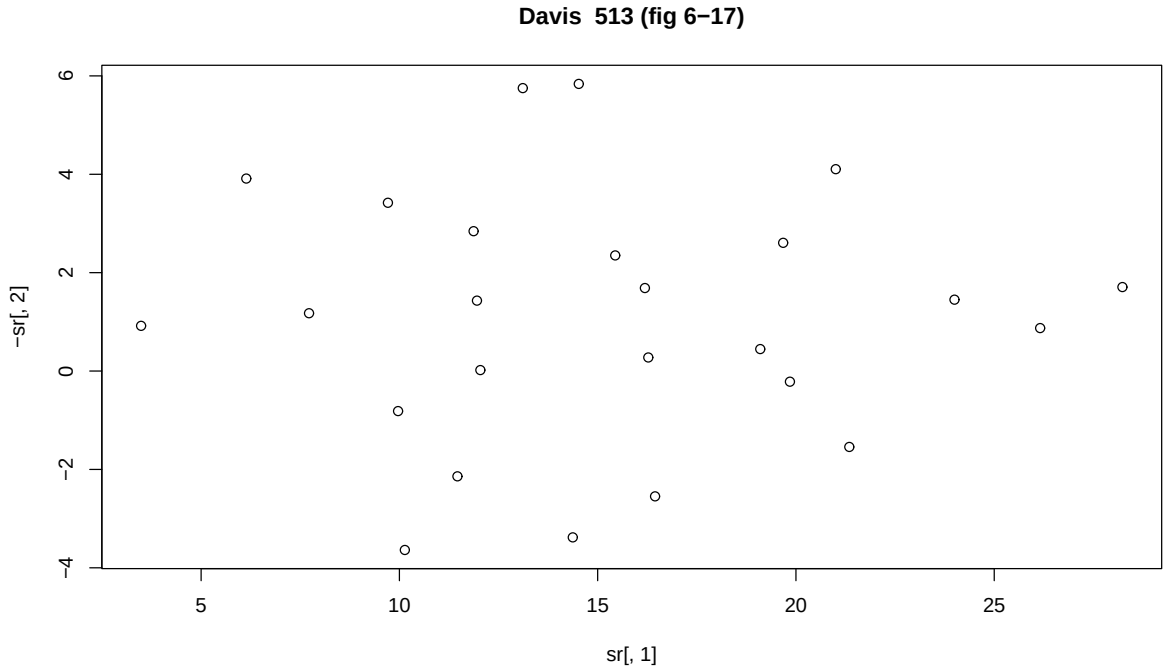

Figure 6: Improved Figure Davis 6-16, page 511

```
> BAR = scan(file = "/home/lees/Class/Data_Analysis/dos_files/BARATARA.TXT",
+     skip = 1, list(t = 0, p1 = 0, p2 = 0, p3 = 0, p4 = 0, p5 = 0,
+         p6 = 0, p7 = 0))
> v = cbind(BAR$p1, BAR$p2, BAR$p3, BAR$p4, BAR$p5, BAR$p6, BAR$p7)
> sv = var(v)
> esv = eigen(sv)
> sr = v %*% esv$vectors
> syms = rep(1, length(BAR$t))
> syms[BAR$t == 2] = 22
> syms[BAR$t == 3] = 5
> syms[BAR$t == 4] = 2
> syms[BAR$t == 5] = 6
> cols = rep(1, length(BAR$t))
> cols[BAR$t == 2] = 2
> cols[BAR$t == 3] = 3
> cols[BAR$t == 4] = 4
> cols[BAR$t == 5] = 5
> v.mean = apply(v, 2, mean)
> h = sweep(v, 2, v.mean)
> tv = (t(h) %*% h)/length(h)
> etv = eigen(tv)
> sr = v %*% esv$vectors
```

```
> postscript(file = "DAVIS522.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(sr[, 1], sr[, 2], asp = TRUE, pch = syms, col = cols, xlab = "PCA 1",
+     ylab = "PCA 2")
> title("Davis  522 (fig 6-24)")
> dev.off()
```

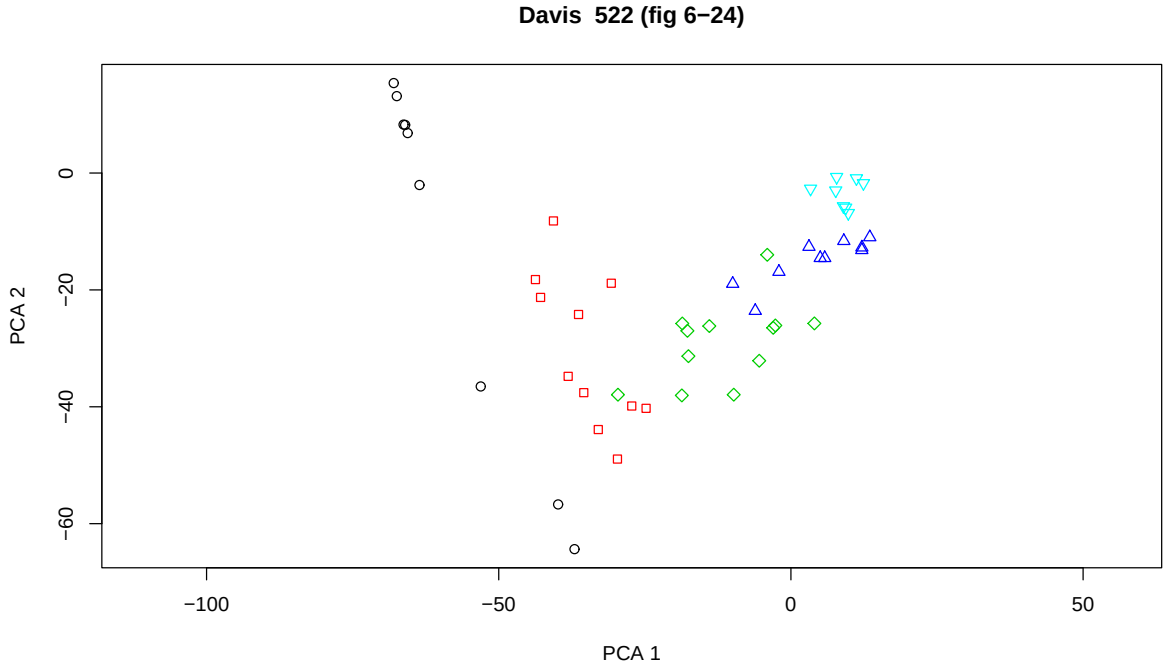

Figure 7: Rotated BARATARA DATA

2.2 Example: WELLWATR.TXT

```
> WW = scan(file = "/home/lees/Class/Data_Analysis/dos_files/WELLWATR.TXT",
+     skip = 1, list(x1 = 0, x2 = 0, x3 = 0, x4 = 0, x5 = 0, n1 = 0))
> w = cbind(WW$x1, WW$x2, WW$x3, WW$x4, WW$x5)
> L = w[WW$n1 == 2, ]
> A = w[WW$n1 == 1, ]
> xlbar = apply(L, 2, mean)
> xabar = apply(A, 2, mean)
```

```
> var(L)
```

```
           [,1]        [,2]       [,3]       [,4]       [,5]
[1,]  5.1614737   0.5133684   7.368316  -1.410316  -3.440211
[2,]  0.5133684  21.0247105  10.694816  -4.089632 -25.397211
[3,]  7.3683158  10.6948158 102.804500 -38.526895 -58.168895
[4,] -1.4103158  -4.0896316 -38.526895  98.865368   7.252000
[5,] -3.4402105 -25.3972105 -58.168895   7.252000 290.870632
```

```
> var(A)
```

```
           [,1]        [,2]       [,3]       [,4]       [,5]
[1,]  5.6394474   0.7332632   8.686763  -2.982158  -4.709500
[2,]  0.7332632  23.1732632  12.765579  -4.559263 -26.987789
[3,]  8.6867632  12.7655789 103.398184 -42.394895 -58.123237
[4,] -2.9821579  -4.5592632 -42.394895 106.952526   9.219947
[5,] -4.7095000 -26.9877895 -58.123237   9.219947 275.261553
```

2.3 Example: OREODONT.TXT

```
> or = scan(file = "/home/lees/Class/Data_Analysis/dos_files/OREODONT.TXT",
+     skip = 1, list(nam = "", a = 0, b = 0, c = 0, d = 0))
> INFO.a = "width of brain case"
> INFO.b = "length of cheek tooth"
> INFO.c = "length of bulla (depression below opening of ear)"
> INFO.d = "depth of bulla"
> cols = match(or$nam, unique(or$nam))
```

```
> postscript(file = "OREODONTplt1.eps", width = 9, height = 6,
+     paper = "special", horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(or$a, or$b, xlab = INFO.a, ylab = INFO.b, col = cols)
> dev.off()
```

```
> postscript(file = "OREODONTplt2.eps", width = 9, height = 6,
+     paper = "special", horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(or$a, or$c, xlab = INFO.a, ylab = INFO.c, col = cols)
> dev.off()
```

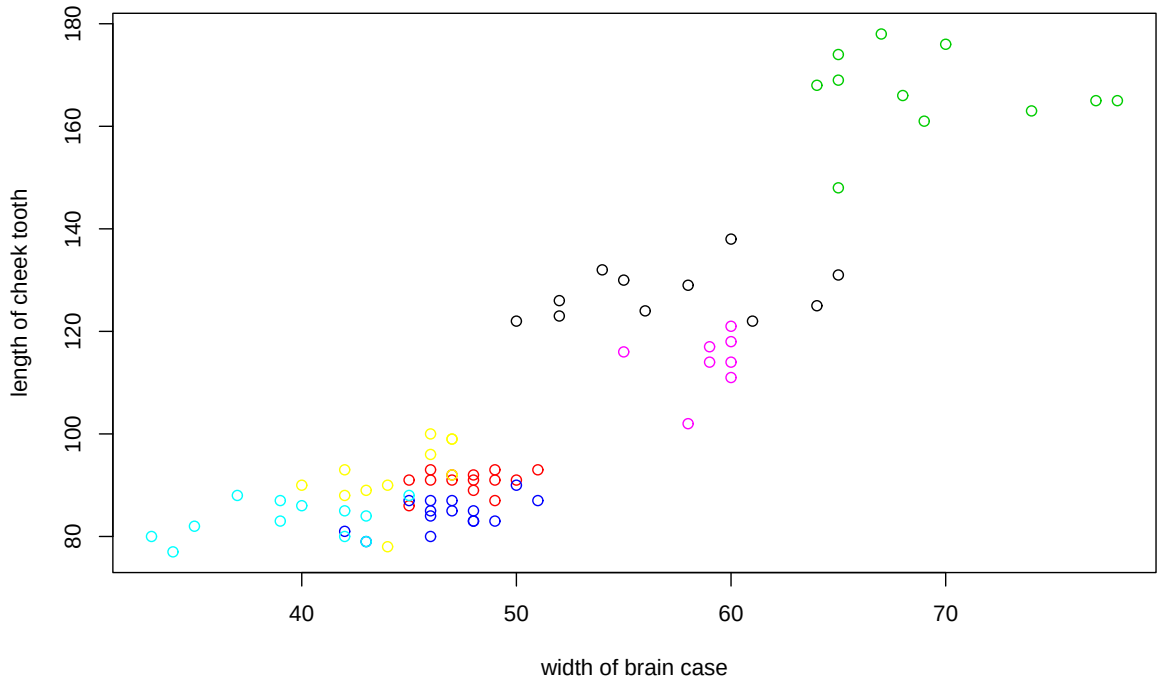

Figure 8: OREODONT DATA, plotted A versus B

```
> postscript(file = "OREODONTplt3.eps", width = 9, height = 6,
+     paper = "special", horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(or$c, or$d, xlab = INFO.c, ylab = INFO.d, col = cols)
> dev.off()
```

Analysis of the Oreodont data,

```
> nms = unique(or$nam)
> A = cbind(or$a, or$b, or$c, or$d)
> va = var(A)
> sv = cor(A)
> esv = eigen(sv)
```

eigenvalues of the correlation matrix are

```
> cat(format(esv$values, digits = 2))
```

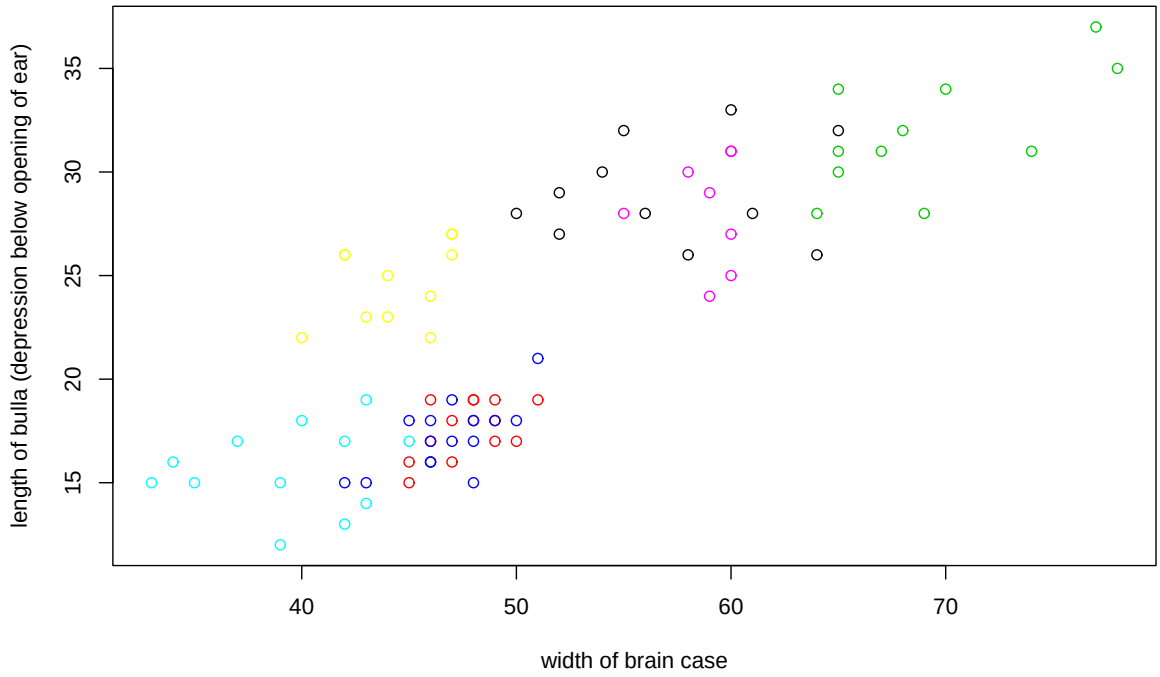

Figure 9: OREODONT DATA, plotted , plotted A versus C

3.444 0.390 0.112 0.055

proportions are

```
> cat(format(esv$values/sum(esv$values), digits = 4))
```

0.86105 0.09741 0.02788 0.01366

```
> postscript(file = "EspectOREO.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(esv$values)
> dev.off()
```

```
> sr = A %*% esv$vectors
> B = cbind(or$a - mean(or$a), or$b - mean(or$b), or$c - mean(or$c),
```

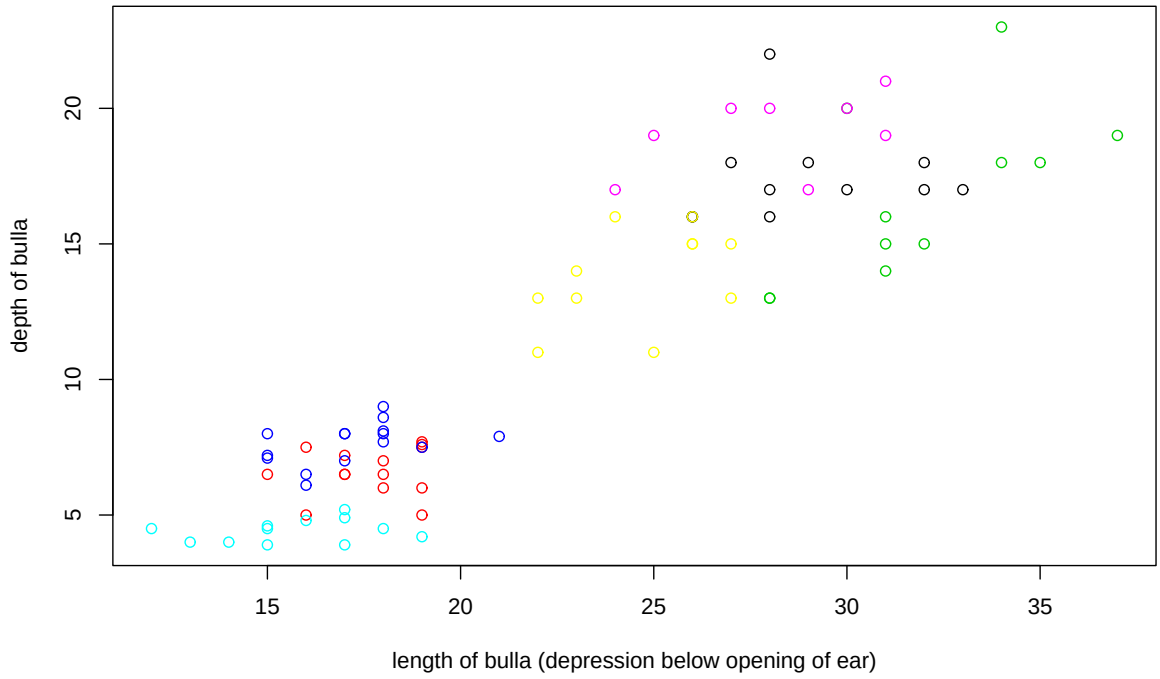

Figure 10: OREODONT DATA, plotted , plotted C versus D

```
+     or$d - mean(or$d))
> enew = esv$vectors
> enew[, 2] = -enew[, 2]
> ga = A %*% enew
```

```
> postscript(file = "OREO_LOADS.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(ga[, 1], ga[, 2], col = cols)
> dev.off()
```

3 Example 3: BOXES.TXT

```
> bx = scan(file = "/home/lees/Class/Data_Analysis/dos_files/BOXES.TXT",
+     skip = 1, list(nam = "", x1 = 0, x2 = 0, x3 = 0, x4 = 0,
```

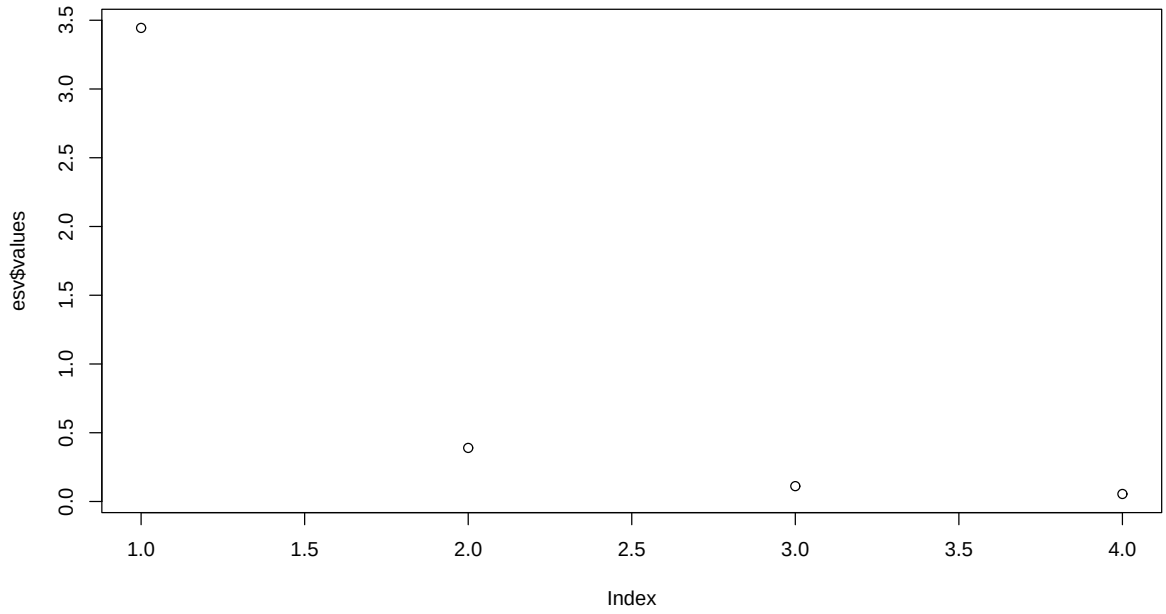

Figure 11: OREODONT: spectrum of eigenvalues

```
+          x5 = 0, x6 = 0, x7 = 0))
> B = cbind(bx$x1, bx$x2, bx$x3, bx$x4, bx$x5, bx$x6, bx$x7)
> v = B
> va = var(B)
> esv = eigen(va)
> ga = B %*% esv$vectors
```

Eigen Values are:

```
> cat(format(esv$values, digits = 2), sep = " ")
```

34.4911 18.9988 2.5388 0.8059 0.3407 0.0334 0.0025

```
> postscript(file = "BOXPCA0.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(esv$values)
> dev.off()
```

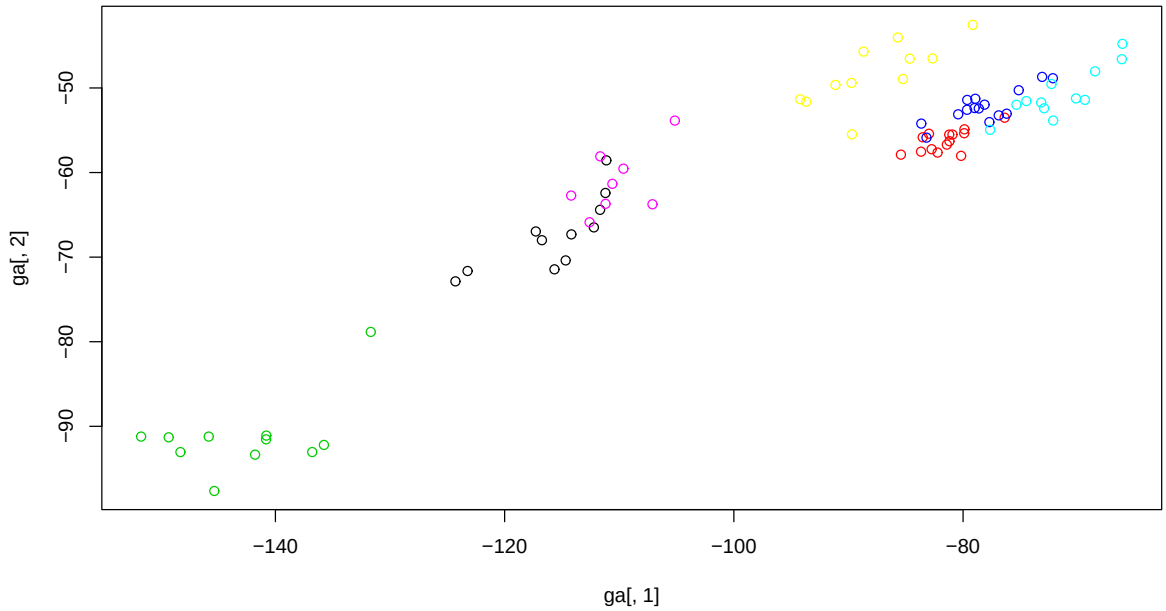

Figure 12: OREODONT LOADINGS

```
> postscript(file = "BOXPCA1.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(ga[, 1], -ga[, 2], type = "n", xlab = "PC1", ylab = "PC2")
> text(ga[, 1], -ga[, 2], labels = bx$nam)
> title("Davis Figure 6-22, p 519")
> dev.off()
```

Set up internal functions and programs

Set up parameters for plotting:

```
> a1 = -45
> i = -(135)
> r1 = rotx(a1)
> r2 = rotz(i)
> R = r2 %*% r1
> Y = range(v[, 1])
> y = 0.4 + (v[, 1] - Y[1])/(Y[2] - Y[1])
> nx = 5
> side = nx
> lentop = length(v[, 1])
```

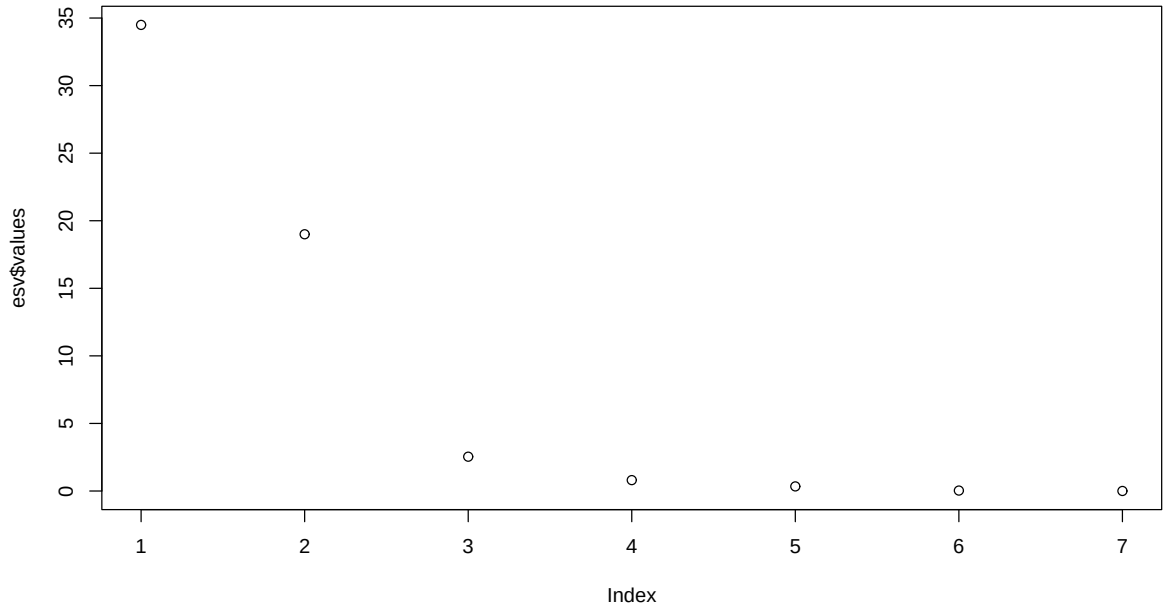

Figure 13: Eigen Spectrum BOXES

Now apply this to put the boxes where they need to go:

```
> postscript(file = "BOXPCA2.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(ga[, 1], -ga[, 2], type = "n", xlab = "PC1", ylab = "PC2")
> for (i in 1:length(v[, 1])) {
+     w = v[i, 1]
+     h = v[i, 3]
+     d = v[i, 2]
+     s = y[i]
+     G = boxglyph(ga[i, 1], -ga[i, 2], s, d, w, h, R)
+     text(ga[i, 1], -ga[i, 2], labels = LETTERS[i], pos = 4)
+ }
> title("Davis Figure 6-22, p 519")
> dev.off()
```

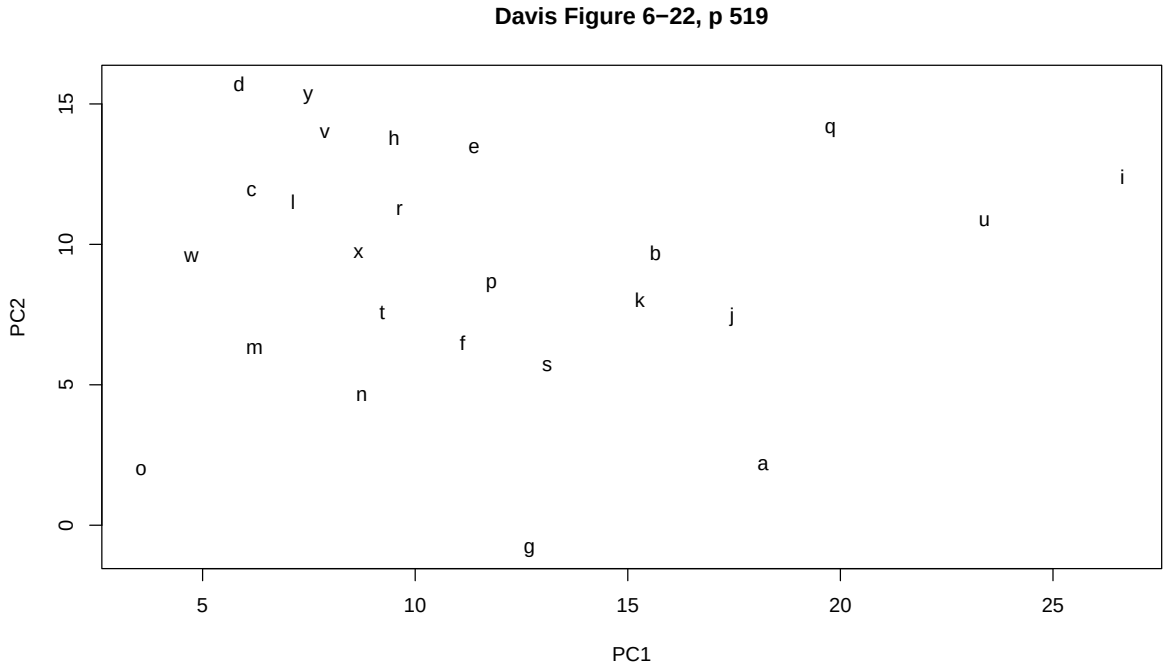

Figure 14: PCA BOXES, Davis Figure 6-22, p 519

4 FACTOR analysis

4.1 SVD

Eckart-Young Theorem For any real matrix $\boldsymbol{X}$ there exists two orthogonal matrices $\boldsymbol{V}$ and $\boldsymbol{U}$ such that,

$$\boldsymbol{V}^T \boldsymbol{X} \boldsymbol{U} = \boldsymbol{\Lambda} \tag{4.1}$$

is real diagonal with no negative values.

4.2 R-mode analysis

```
> gon = matrix(c(4, 27, 18, 12, 25, 12, 10, 23, 16, 14, 21, 14),
+     ncol = 3, byrow = TRUE)
> m.gon = apply(gon, 2, mean)
> agon = sweep(gon, 2, m.gon)
> X = agon
> R = t(agon) %*% agon
```

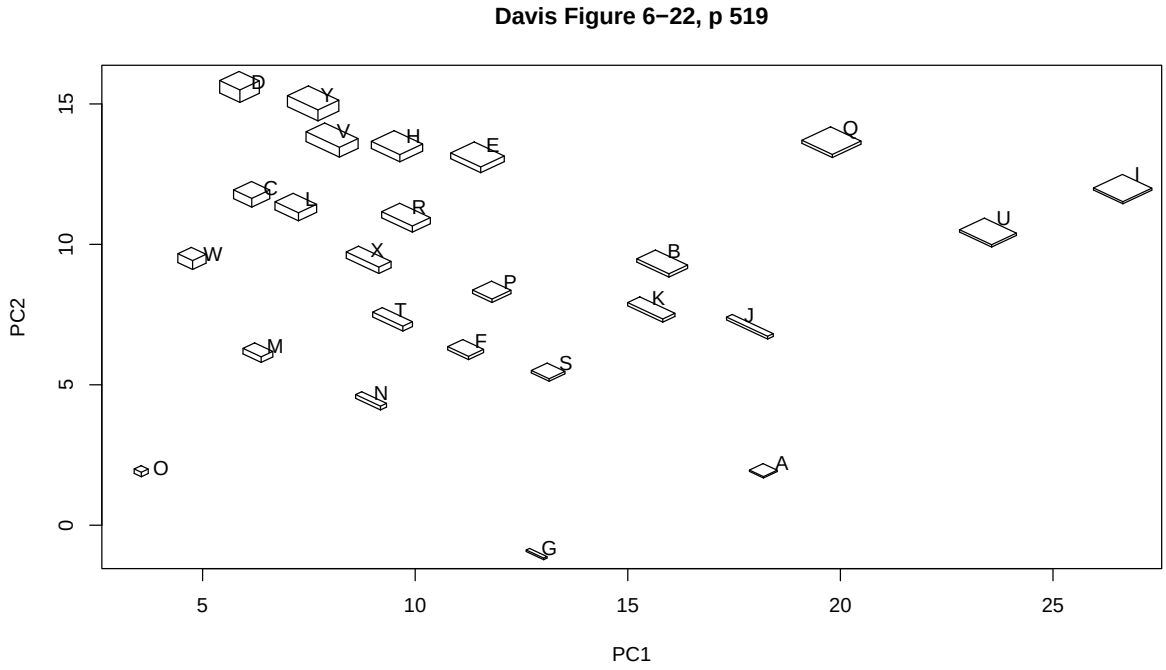

Figure 15: PCA BOXES, Davis Figure 6-22, p 519

```
> egon = eigen(R)
> LAM2 = diag(egon$values[c(1, 2)])
> LAM = sqrt(LAM2)
> sgon = svd(agon)
> U = sgon$v
> U = sgon$v[, c(1, 2)]
> U[, 2] = -U[, 2]
> Ar = U %*% LAM
> score1 = agon %*% Ar
```

```
> postscript(file = "fac_scor_1.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> plot(score1[, 1], score1[, 2])
> text(score1[, 1], score1[, 2], labels = LETTERS[1:4], pos = 3)
> dev.off()
```

4.3 Q-mode analysis

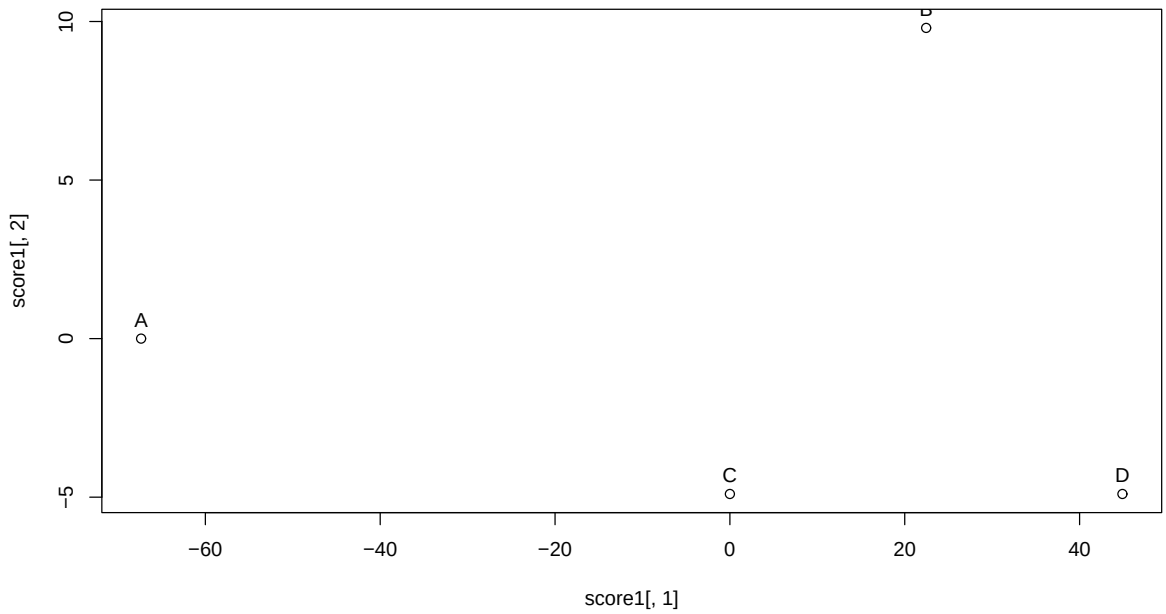

Figure 16: Factor Scores

```
> Q = (agon) %*% t(agon)
> V = sgon$u
> V = sgon$u[, c(1, 2)]
> V[, 2] = -V[, 2]
> Aq = V %*% LAM
> Sq = t(X) %*% Aq
> Ar %*% LAM
```

```
          [,1]          [,2]
[1,]  68.58571 -9.621933e-16
[2,] -34.29286  8.485281e+00
[3,] -34.29286 -8.485281e+00
```

```
> V %*% LAM %*% t(U)
```

```
              [,1] [,2] [,3]
[1,] -6.000000e+00    3    3
[2,]  2.000000e+00    1   -3
[3,]  8.722736e-18   -1    1
[4,]  4.000000e+00   -3   -1
```

```
> plot(Sq[, 1], Sq[, 2])
> abline(v = 0, h = 0, lty = 2, col = grey(0.9))
> text(Sq[, 1], Sq[, 2], labels = 1:3, pos = 3)
> arrows(rep(0, 3), rep(0, 3), Sq[, 1], Sq[, 2])
```

APPENDIX

A BOXES functions

```
> boxglyph <- function(x, y, s, d, w, h, R) {
+     A = sqrt(d^2 + w^2 + h^2)
+     d = d/A
+     w = w/A
+     h = h/A
+     mat = matrix(c(0, 0, h, d, 0, h, d, w, h, 0, w, h, d, 0,
+         0, d, w, 0, 0, w, 0), ncol = 3, byrow = TRUE)
+     S = diag(s, ncol = 3, nrow = 3)
+     g = mat %*% R %*% S
+     g[, 1] = g[, 1] + x
+     g[, 2] = g[, 2] + y
+     lines(g[c(1:4, 1), 1], g[c(1:4, 1), 2])
+     segments(g[c(2, 3, 4), 1], g[c(2, 3, 4), 2], g[c(5, 6, 7),
+         1], g[c(5, 6, 7), 2])
+     segments(g[c(5, 6), 1], g[c(5, 6), 2], g[c(6, 7), 1], g[c(6,
+         7), 2])
+     invisible(g)
+ }
> rotx <- function(deg) {
+     rad1 = deg * 0.0174532
+     r = diag(3)
+     r[2, 2] = cos(rad1)
+     r[2, 3] = sin(rad1)
+     r[3, 3] = r[2, 2]
+     r[3, 2] = -r[2, 3]
+     return(r)
+ }
> roty <- function(deg) {
+     rad1 = deg * 0.0174532
+     r = diag(3)
```

```
+     r[1, 1] = cos(rad1)
+     r[3, 1] = sin(rad1)
+     r[3, 3] = r[1, 1]
+     r[1, 3] = -r[3, 1]
+     return(r)
+ }
> rotz <- function(deg) {
+     rad1 = deg * 0.0174532
+     r = diag(3)
+     r[1, 1] = cos(rad1)
+     r[1, 2] = sin(rad1)
+     r[2, 2] = r[1, 1]
+     r[2, 1] = -r[1, 2]
+     return(r)
+ }
```

```
> a1 = -45
> i = -(135)
> r1 = rotx(a1)
> r2 = rotz(i)
> R = r2 %*% r1
> Y = range(v[, 1])
> y = 0.4 + (v[, 1] - Y[1])/(Y[2] - Y[1])
> nx = 5
> side = nx
> lentop = length(v[, 1])
```

Create the plot with the boxes:

```
> postscript(file = "BOXES.eps", width = 10, height = 6, paper = "special",
+     horizontal = FALSE, onefile = TRUE, print.it = FALSE)
> par(mai = c(0, 0, 0, 0))
> plot(c(0, 2 * (nx + 1)), c(0, 2 * (nx + 1)), asp = TRUE, type = "n",
+     axes = FALSE, xlab = "", ylab = "")
> for (i in 1:length(v[, 1])) {
+     iy = floor((i - 1)/side) + 1
+     ix = i - (iy - 1) * nx
+     if (ix == 0) {
+         ix = nx
+     }
+     ix = 2 * ix
```

```
+     iy = 2 * iy
+     w = v[i, 1]
+     h = v[i, 3]
+     d = v[i, 2]
+     s = y[i]
+     G = boxglyph(iy, ix, s, d, w, h, R)
+     text(G[4, 1], G[4, 2], labels = LETTERS[i], pos = 4)
+ }
> dev.off()
```

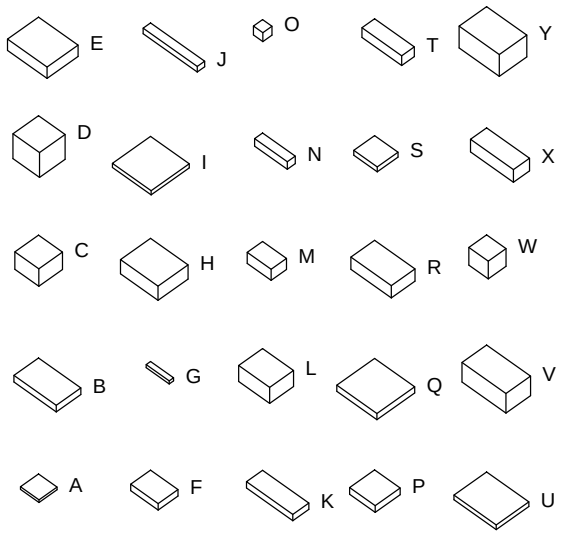

Figure 17: PCA BOXES, Davis Figure 6-22, p 519