

## Lab 12: Dimension reduction in R

June 6, 2003

In this lab, we present some dimension reduction algorithms that are available in R and that are useful for classification of microarray data. The first example in this section will rely on the data reported in Hedenfalk et al. (2001) where one of the goals is to find genes that are differentially expressed between BRCA1-mutation-positive tumors and BRCA2-mutation-positive tumors by obtaining several microarrays from each cell type.

We first load the necessary libraries for this lab.

```
> library(mda)
> library(dr)
> library(Milan)
```

Expression measures are stored in an `hedenfalk` object available through the `brca` package.

```
> data(brca)
> names(brca) = c("plate", "cid", "BC11", "BC15", "BC13", "BC17",
+ "BC12", "BC14", "BC210a", "BC29", "BC28", "BC210b", "SP16",
+ "SP17", "SP15", "SP18", "SP19", "SP21", "SP20", "BC16",
+ "BC213", "BC214", "BC211", "BC212")
> brca <- (brca[, c(-1, -2)])
> brca.class <- c(rep(1, 6), rep(2, 4), rep(0, 7), 1, rep(2, 4))
```

We formulate the effects of gene expression on class type using the *multinomial logistic regression model*:

$$\log \frac{P(Y_i = r)}{P(Y_i = 0)} = \mathbf{X}_i \cdot \beta_{r0}, \quad r = 1, \dots, G - 1,$$

where  $\beta_{r0}$  is a  $p$ -dimensional vector of unknown regression coefficients. Since  $p \gg n$  it is not possible to estimate the parameters of the above model using standard statistical methods. A principal component study becomes then a suitable first step to reduce the dimension of  $\beta_{r0}$ .

We will first apply a singular value decomposition based logistic regression method to classify the cells. It is much like principal component regression analysis (PCR). In order to setup the model we define first the design matrix for discrimination.

```

> brcam = matrix(brca)
> brda = as.vector(brca$BC11[[1]])
> for (i in 2:22) brda = cbind(brda, brcam[[i]][[1]])

```

We now apply the SVD logistic procedure, made by Ghosh (2001) and available in the *Milan*.

```

> da1 <- svdrfda(factor(brca.class) ~ t(brda), K = 3)
> plot.fda(da1)
> confusion(predict(da1), factor(brca.class))

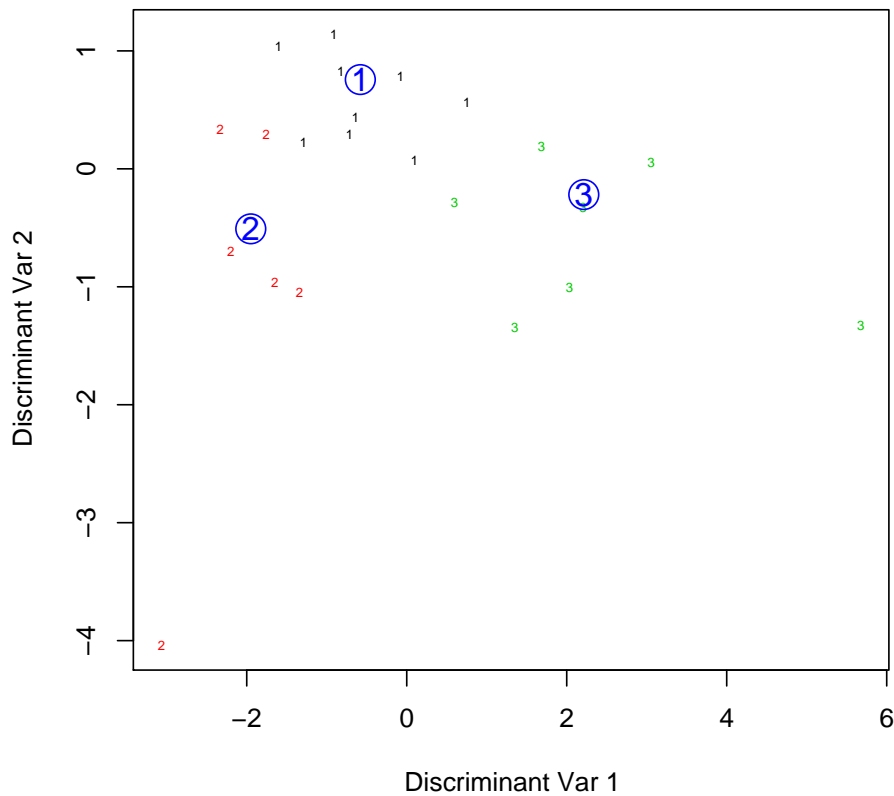
```

```

      true
object 0 1 2
      0 7 1 1
      1 0 6 0
      2 0 0 7
attr(,"error")
[1] 0.0909091

```

**Discriminant Plot for predict classes**



SVD produces orthogonal class descriptors that reduce the high dimensional data (supergenes). This is achieved without regards to the response variation and may be inefficient. This way of reducing the regressor dimensionality is totally independent of the output variable. Another method is PLS, where the PLS components are chosen so that the sample covariance between the response and a linear combination of the  $p$  predictors (genes) is maximum.

```
> da2 <- svdpls1fda(factor(brca.class) ~ t(brda), K = 3)
```

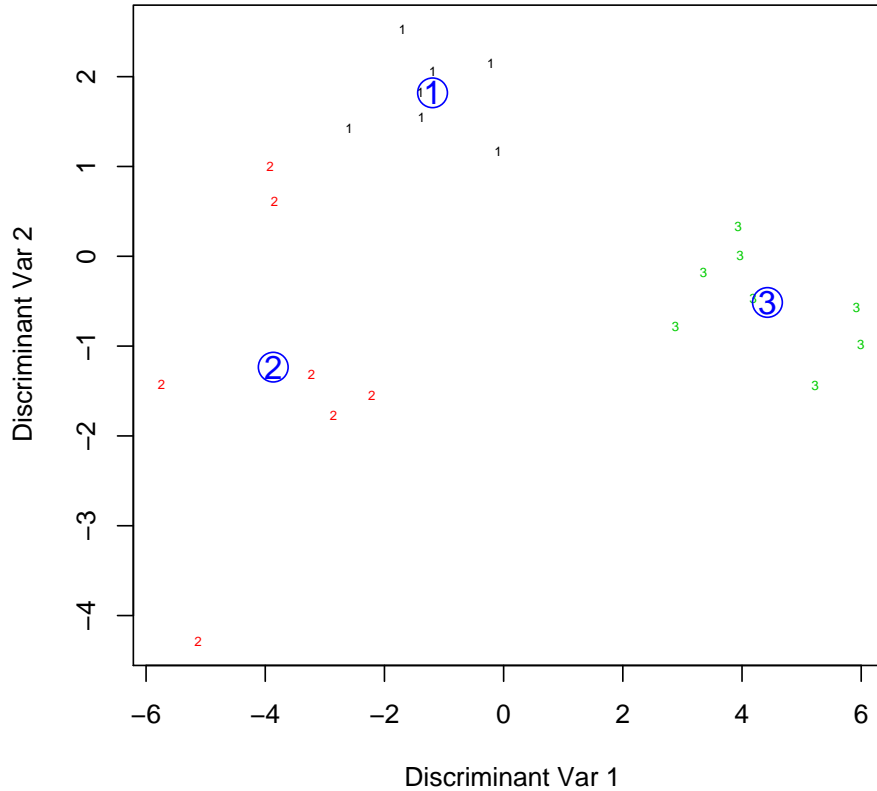
```
2
```

```
> plot.fda(da2)
```

```
> confusion(predict(da1), factor(brca.class))
```

```
      true
object 0 1 2
      0 7 1 1
      1 0 6 0
      2 0 0 7
attr(,"error")
[1] 0.0909091
```

**Discriminant Plot for predict classes**



However, PLS is really designed to handle continuous responses and especially for models that do not really suffer from conditional heteroscedasticity as it is the case for binary or multinomial data, as it is the case here. An extension of the standard PLS algorithm to Generalized linear Models is also available from Marx. Here is an example on how to use such a procedure.

```
> y <- c(rep(1, 6), rep(0, 11), 1, rep(0, 4))
> fit <- gpls(y, t(brda), components = 4, family = "binomial",
+   link = "logit")
```

```
[1] 1 1 0
[1] 2.000 2.375 1.000
[1] 3.0000000 0.3648937 0.1548246
[1] 4.0000000 0.2831986 0.5233251
[1] 5.0000000 0.2104281 21.4216001
[1] 6.0000000 0.1629339 1.0425011
[1] 7.0000000 0.1504187 3.7663149
[1] 8.0000000 0.3316823 0.8699909
```

```

[1] 9.0000000 0.2246022 1.7113645
[1] 10.0000000 0.1381122 3.8310665
[1] 11.0000000 0.4059827 0.6957843
[1] 12.0000000 0.3065472 0.3682269
[1] 13.0000000 0.1582838 0.5851757
[1] 14.0000000 0.2909088 0.6314775
[1] 15.0000000 0.2259065 0.6658327
[1] 16.0000000 0.4839746 1.9946585
[1] 17.0000000 0.1014345 26.6008493
[1] 18.0000000 0.3087664 22.1390652
[1] 19.0000000 0.9120893 3.9259203
[1] 20.0000000 0.344829 3.189400
[1] 21.0000000 0.2626732 1.4324321
[1] 22.0000000 0.1394335 0.9186048
[1] 23.0000000 1.452462 2.811113
[1] 24.0000000 0.62279 16.24731
[1] 25.0000000 0.2867805 5.2564918

```

```

> z = fit$mu > 0.5
> z = as.integer(z)
> confusion(z, y)

```

```

      true
object 0 1
      0 15 0
      1 0 7
attr(,"error")
[1] 0

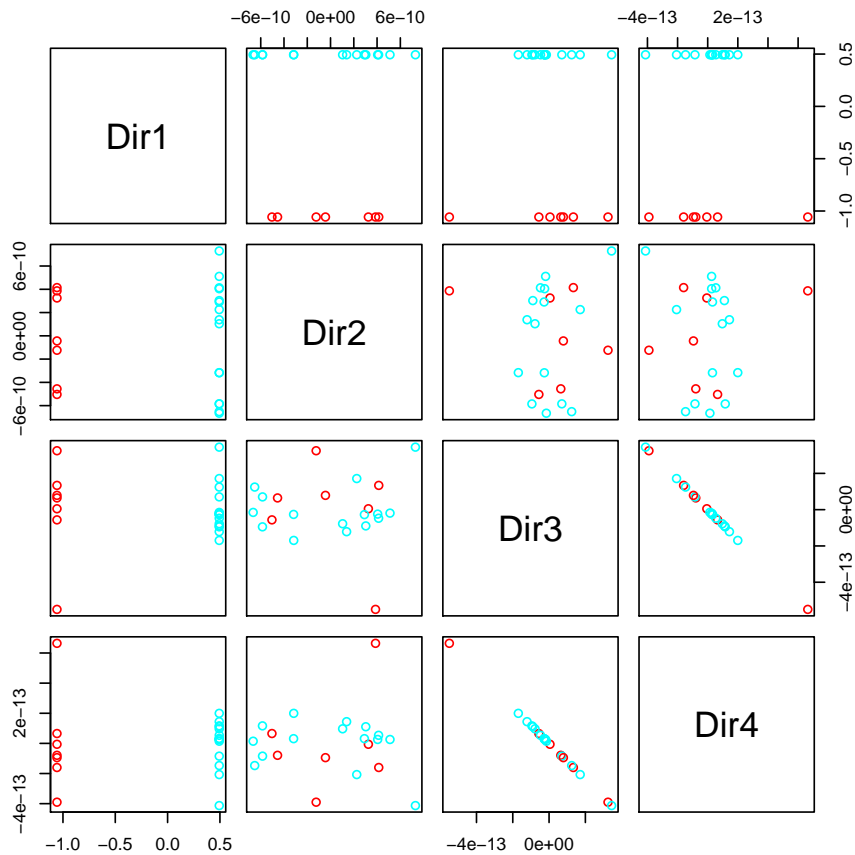
```

The purpose of the following commands is to apply other type of dimension reduction methods to the BRCA data. They are based on sliced inverse regression methods such as those implemented in *dr*. The following commands estimate the central dimension reduction subspace and perform a two-class discrimination. Note that since we are dealing with binary responses the number of slices must be two.

```

> fitdr = dr(y ~ t(brda), method = "Sir", nslices = 2)
> plot(fitdr, mark.by.y = T)

```



When  $X$  are normally distributed, SIR is equivalent to Linear Discriminant Analysis in the sense that they both estimate the same discriminant linear combinations of the predictors and SAVE is equivalent to Quadratic Discriminant Analysis.

```
> fitdr = dr(y ~ t(brda), method = "Save", nslices = 2)
> dr.permutation.test(fitdr, npermute = 100)
```

Permutation tests

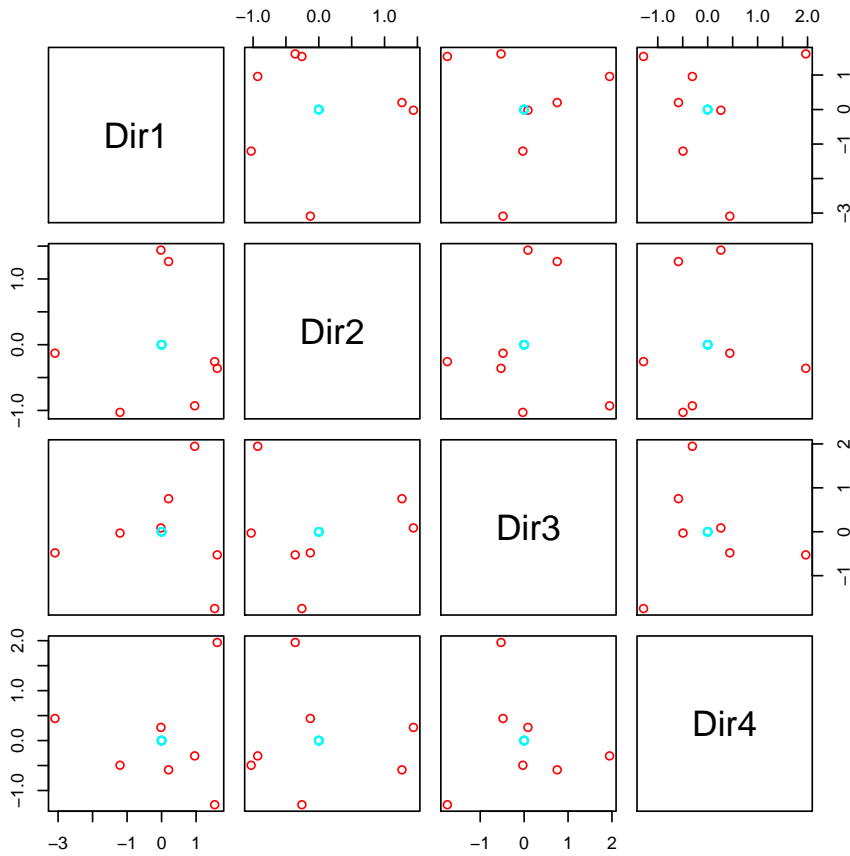
Number of permutations:

```
[1] 100
```

Test results:

	Stat	p-value
0D vs >= 1D	1237.2	0.9802
1D vs >= 2D	1172.5	0.7921
2D vs >= 3D	1107.7	0.7228
3D vs >= 4D	1042.9	0.8614
4D vs >= 5D	978.1	0.9406

```
> plot(fitdr, mark.by.y = T)
```



We have applied sliced inverse regression methods to binary data but note that SIR and SAVE can also be applied to problems with multinomial or multi-valued responses.