Supervised Dimension Reduction

S. Tonini, F. Chiaromonte (special thanks to J. Di Iorio, L. Insolia and L. Testa)

March 13th 2025

Introduction

Libraries

We are going to use:

```
library(tidyverse)  # data manipulation and visualization
library(plotly)     # plots in 3D
library(ggplot2)    # plots in 2D
library(ggpubr)     # to combine multiple ggplot objects (ggarrenge)
library(mvtnorm)    # to generate multivariate normal distribution
library(dr)         # SIR
library(factoextra) # PCA-related functions
```

Data

Let’s first define a function to generate Gaussian data. This function takes four arguments:

- n: number of observations;
- center: the mean vector
- sigma: the covariance matrix
- label: the cluster label

```
generateGaussianData <- function(n, center, sigma, label) {
  data = rmvnorm(n, center, sigma)
  data = data.frame(data)
  names(data) = c("x", "y", "z")
  data = data %>% mutate(class=factor(label))
  data
}
```

Now let’s simulate a dataset.

```r
covmat <- matrix(c(1,0.88,0.88,0.88, 1,0.88,0.88,0.88, 1),
       nrow = 3, byrow=T)

# cluster 1
n = 200
center = c(2, 8, 6)
sigma = covmat
group1 = generateGaussianData(n, center, sigma, 1)

# cluster 2
n = 200
center = c(4, 8, 6)
sigma = covmat
group2 = generateGaussianData(n, center, sigma, 2)

# cluster 3
n = 200
center = c(6, 8, 6)
sigma = covmat
group3 = generateGaussianData(n, center, sigma, 3)

# all data
df = bind_rows(group1, group2, group3)

head(df)
```

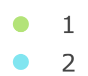
```
##          x        y        z class
## 1 2.307499 7.085100 6.232529     1
## 2 1.897781 8.343378 6.250728     1
## 3 2.076826 8.150770 5.934447     1
## 4 2.540832 7.858882 6.702368     1
## 5 2.468963 8.019310 6.667697     1
## 6 1.626900 7.558433 5.640764     1
```

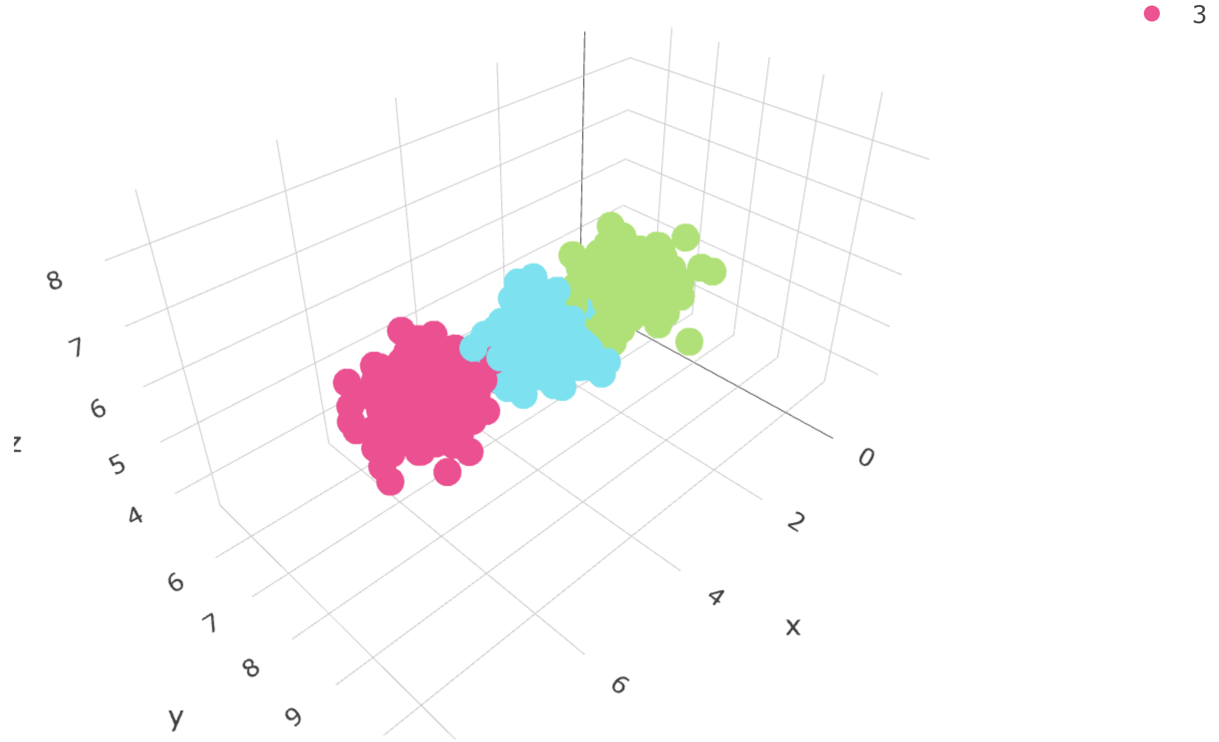
```r
summary(df)
```

```
##        x                  y                z          class
##  Min.   :-0.9245   Min.   : 5.277   Min.   :3.491   1:200
##  1st Qu.: 2.4281   1st Qu.: 7.288   1st Qu.:5.268   2:200
##  Median : 4.0780   Median : 8.054   Median :5.964   3:200
##  Mean   : 4.0000   Mean   : 8.003   Mean   :5.967
##  3rd Qu.: 5.6639   3rd Qu.: 8.724   3rd Qu.:6.681
##  Max.   : 8.3003   Max.   :10.761   Max.   :8.609
```

And plot our simulated data.

```r
fig <- plot_ly(df, x = ~x, y = ~y, z = ~z,
               color = ~class, colors = c('#b3e378', '#81e5f0', '#ed5391'))
fig <- fig %>% add_markers()
fig <- fig %>% layout(scene = list(xaxis = list(title = 'x'),
                                   yaxis = list(title = 'y'),
                                   zaxis = list(title = 'z')))
fig
```





PCA vs LDA

PCA

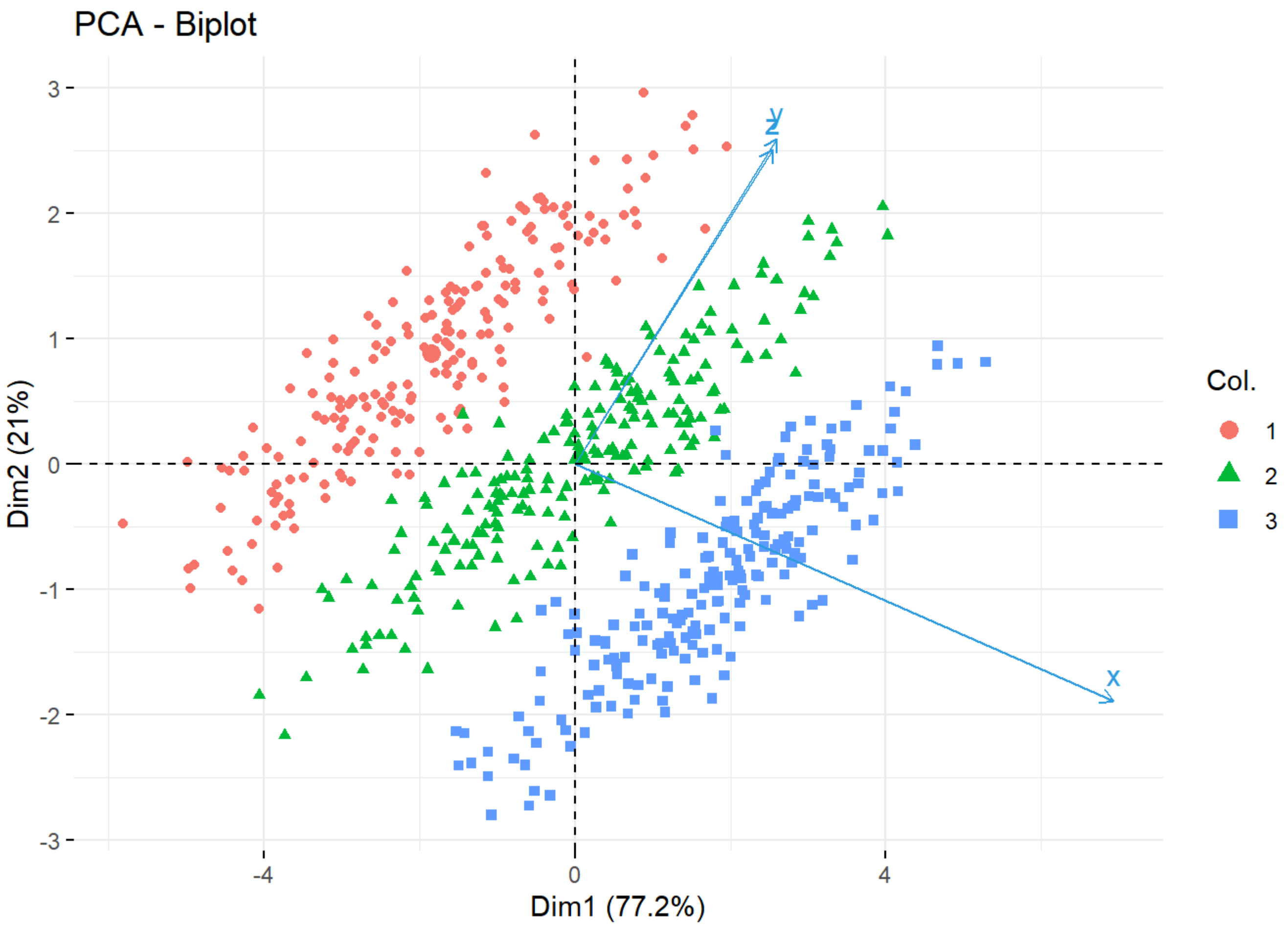
Now let us perform PCA.

```
pc <- prcomp(df[,c(1,2,3)])
get_eig(pc)
```

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1  4.5072810        77.179051                    77.17905
## Dim.2  1.2251734        20.978883                    98.15793
## Dim.3  0.1075772         1.842066                   100.00000
```

This is the corresponding biplot.

```
fviz_pca_biplot(pc, col.var= "#2E9FDF", col.ind= df$class, label="var")
```



Note that considering the first two principal components it is impossible to notice differences within the three groups (all groups are overlapping).

LDA

Let's perform LDA:

```
lda.df <- lda(factor(class) ~ x + y + z, data = df)
lda.df
```

```
## Call:
## lda(factor(class) ~ x + y + z, data = df)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3333333 0.3333333 0.3333333
##
## Group means:
##          x        y        z
## 1 1.954932 7.944171 5.914553
## 2 4.026109 8.063209 6.013852
## 3 6.018984 8.000274 5.972194
##
## Coefficients of linear discriminants:
##         LD1        LD2
## x  2.406106  0.0120225
## y -1.082677 -1.3795532
## z -1.196362  0.4756419
##
## Proportion of trace:
##    LD1    LD2
## 0.9999 0.0001
```

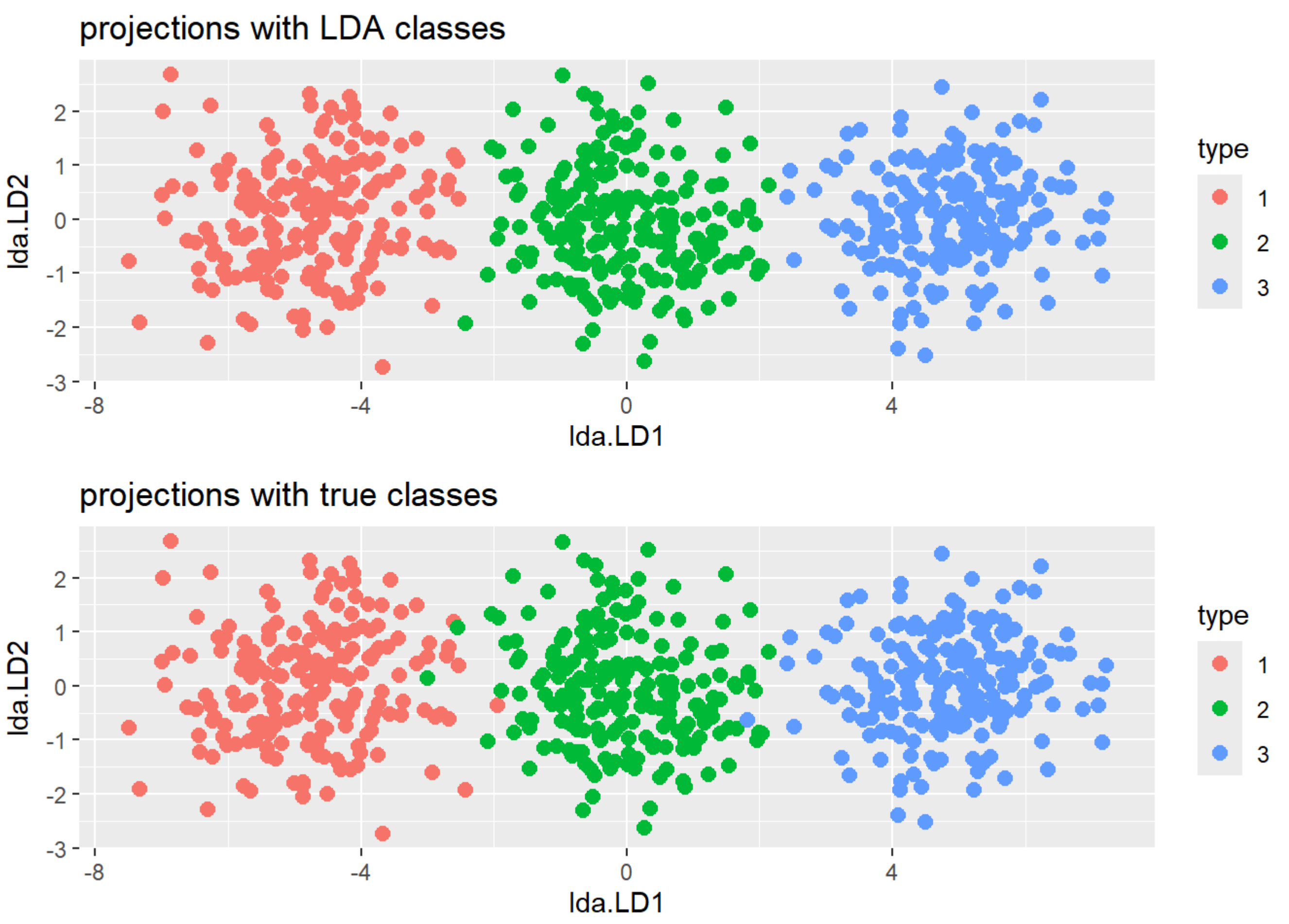
Let us plot the projections on LD1 and LD2

```
# prediction on df to get projections
predmodel.lda = predict(lda.df, data=df)

# projections with LDA classes
estclass <- as.factor(apply(predmodel.lda$posterior, 1, which.max))
newdata2 <- data.frame(type = estclass, lda = predmodel.lda$x)
p1 <- ggplot(newdata2) +
        geom_point(aes(lda.LD1, lda.LD2, colour = type), size = 2.5) +
        ggtitle("projections with LDA classes")

# projections with true classes
newdata <- data.frame(type = df$class, lda = predmodel.lda$x)
p2 <- ggplot(newdata) +
        geom_point(aes(lda.LD1, lda.LD2, colour = type), size = 2.5) +
        ggtitle("projections with true classes")


ggarrange(p1,p2,
          nrow=2)
```



SIR

Now we use the SIR (Sliced Inversion Regression) in the dr package

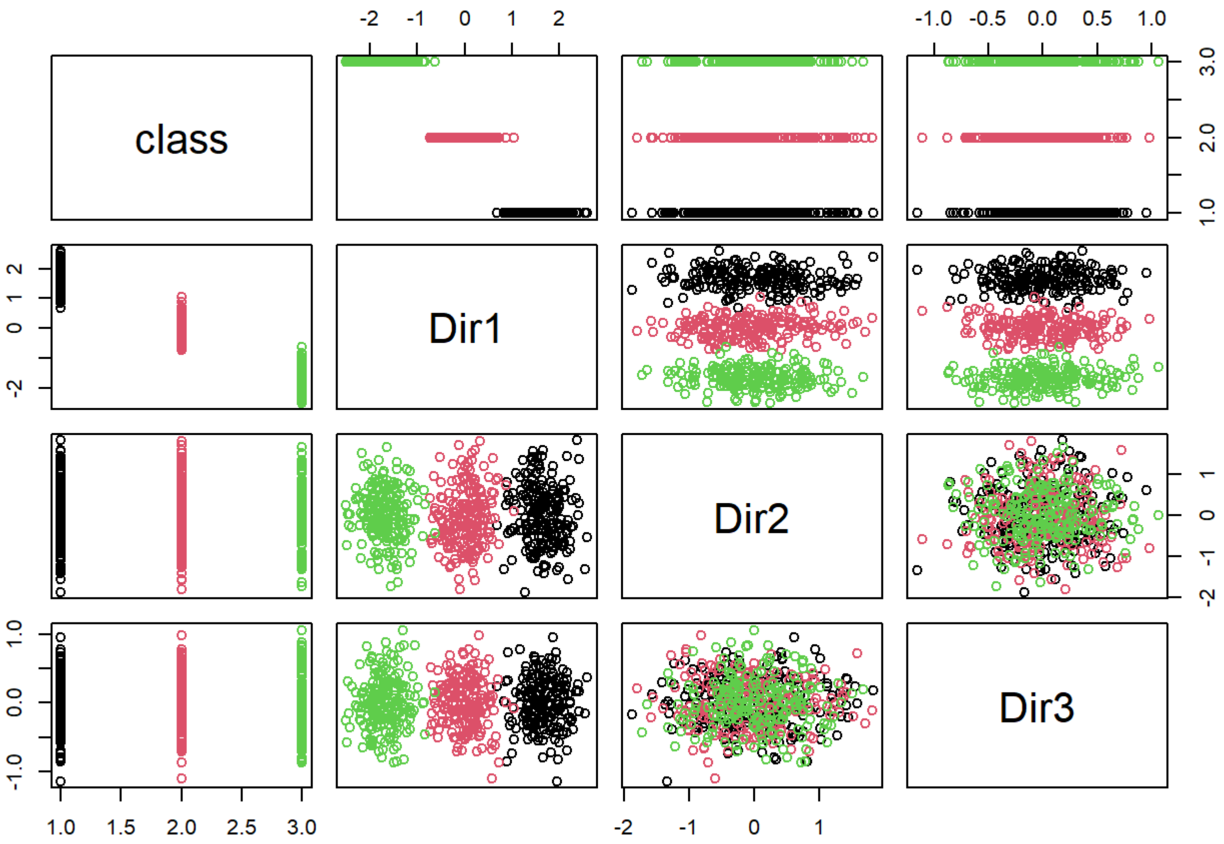
```
# default fitting method is "sir"
help(dr)

dr_res <- dr(class ~ x + y + z, data = df, method='sir')

dr_res
```

```
##
##  dr(formula = class ~ x + y + z, data = df, method = "sir")
## Estimated Basis Vectors for Central Subspace:
##         Dir1        Dir2         Dir3
## x  0.8305406 -0.00823856 -0.002777379
## y -0.3737188  0.94535485 -0.611629599
## z -0.4129608 -0.32593915  0.791139381
## Eigenvalues:
## [1] 9.397460e-01 1.867652e-03 8.950843e-18
```

```
plot(dr_res, col=df$class)
```

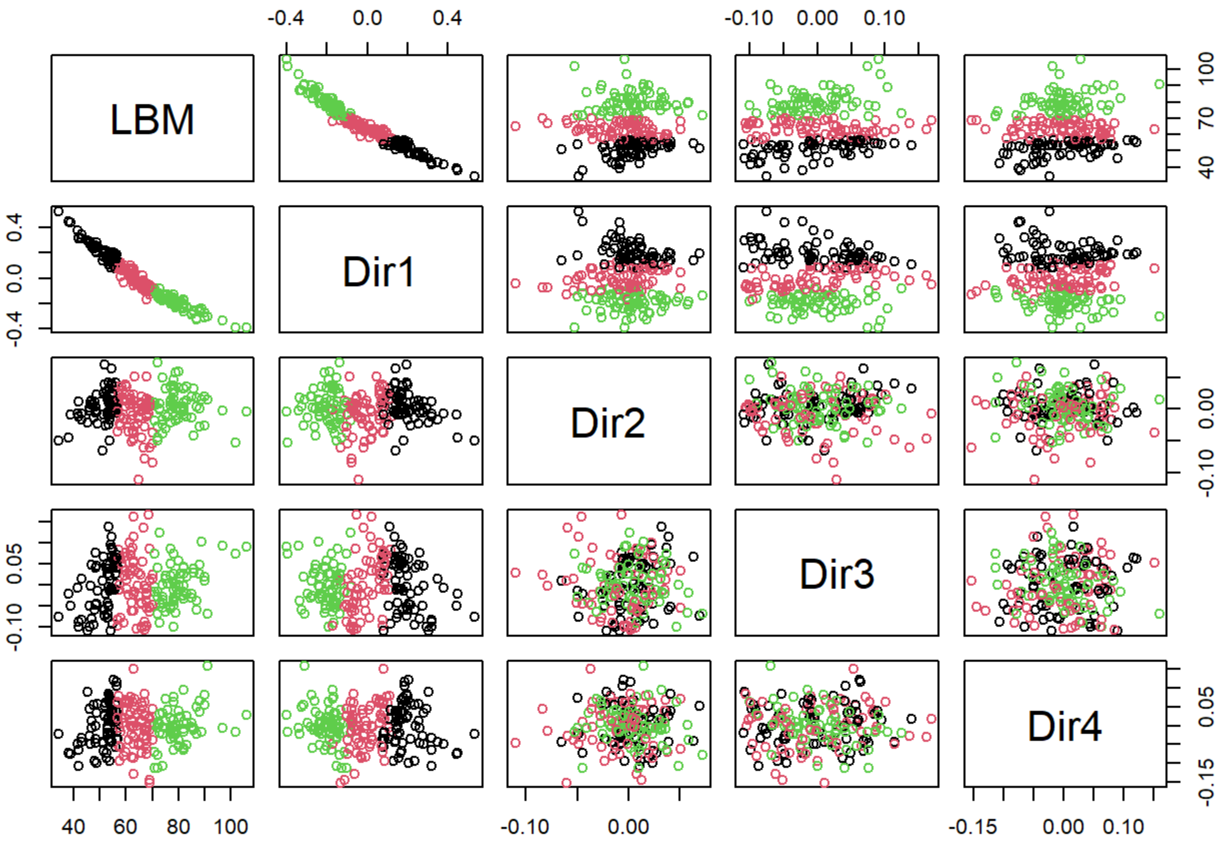



```
names(dr_res)
```

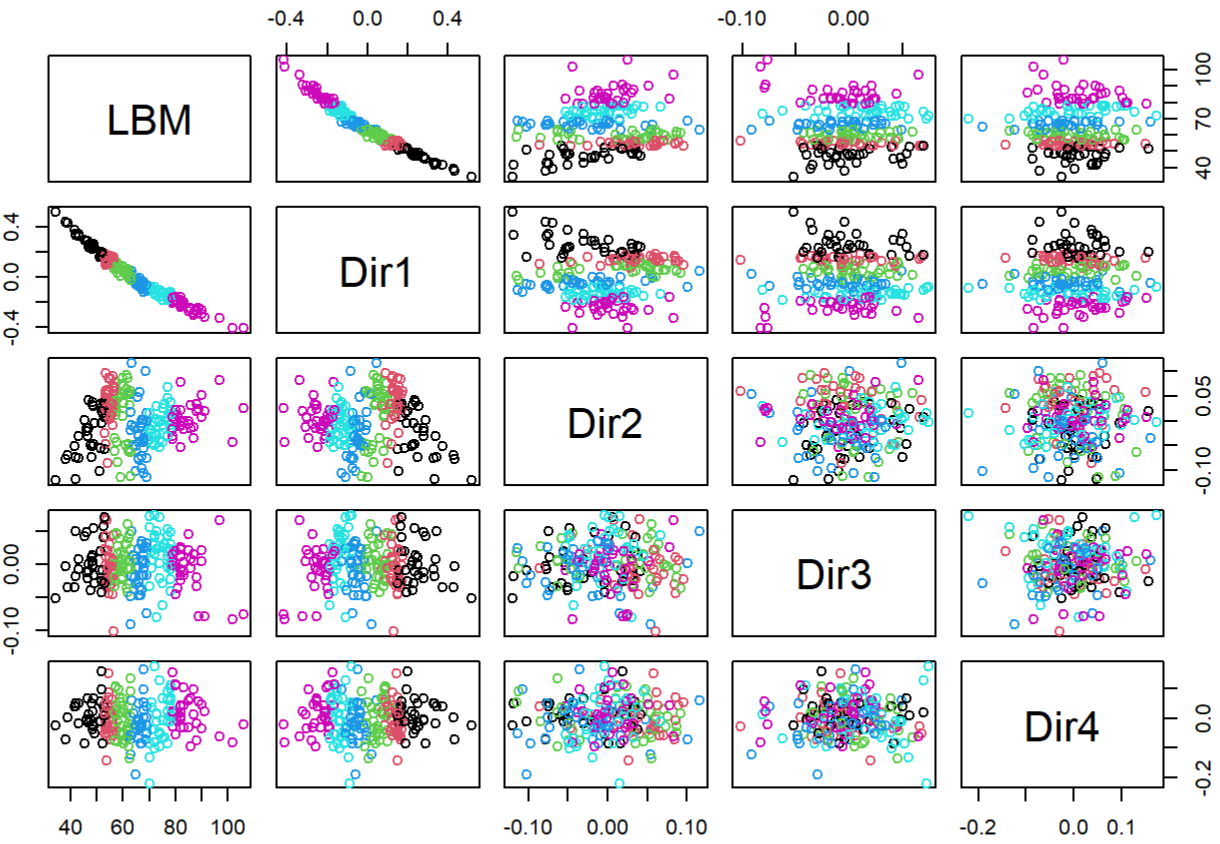
```
##  [1] "x"            "y"            "weights"      "method"       "cases"
##  [6] "qr"           "group"        "chi2approx"   "evectors"     "evalues"
## [11] "numdir"       "raw.evectors" "M"            "slice.info"   "call"
## [16] "y.name"       "terms"
```

We perform SIR on real data with continuos outcome

```
data(ais)
?ais
dr_res3 <-dr(LBM~log(SSF)+log(Wt)+log(Hg)+log(Ht)+log(WCC)+log(RCC)+
               log(Hc)+log(Ferr),data=ais, nslices=3)
plot(dr_res3, col=dr_res3$slice.info$slice.indicator)
```




```
dr_res6 <-dr(LBM~log(SSF)+log(Wt)+log(Hg)+log(Ht)+log(WCC)+log(RCC)+
               log(Hc)+log(Ferr),data=ais, nslices=6)
plot(dr_res6, col=dr_res6$slice.info$slice.indicator)
```


LBM
Dir1
Dir2
Dir3
Dir4
-0.4
0.0
0.4
-0.10
0.00
100
70
40
0.05
-0.10
0.0
-0.2
40
60
80
100
0.10
-0.2
0.1