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.

```
covmat <- diag(3)

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

# cluster 2
n = 200
center = c(12, 8, 6)
sigma = covmat
group2 = generateGaussianData(n, center, sigma, 2)

# cluster 3
n = 200
center = c(22, 8, 6)
sigma = covmat
group3 = generateGaussianData(n, center, sigma, 3)

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

head(df)
```

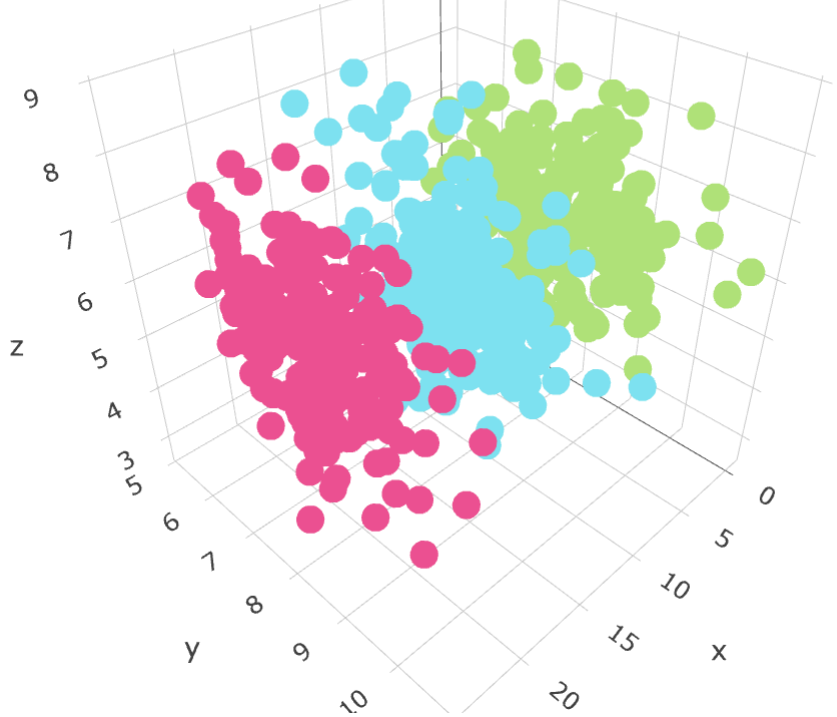
```
##          x        y        z class
## 1 1.088204 8.564831 7.444229     1
## 2 1.580120 5.767798 6.240231     1
## 3 1.244599 7.645695 5.526186     1
## 4 3.503039 7.300759 6.219655     1
## 5 3.222110 8.449626 4.808901     1
## 6 1.809090 7.539433 4.822294     1
```

```
summary(df)
```

```
##        x                y                z         class
##  Min.   :-0.621   Min.   : 5.139   Min.   :2.674   1:200
##  1st Qu.: 2.489   1st Qu.: 7.379   1st Qu.:5.379   2:200
##  Median :12.083   Median : 7.966   Median :6.046   3:200
##  Mean   :11.991   Mean   : 7.971   Mean   :6.062
##  3rd Qu.:21.186   3rd Qu.: 8.596   3rd Qu.:6.677
##  Max.   :24.777   Max.   :10.950   Max.   :8.872
```

And plot our simulated data.

```
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

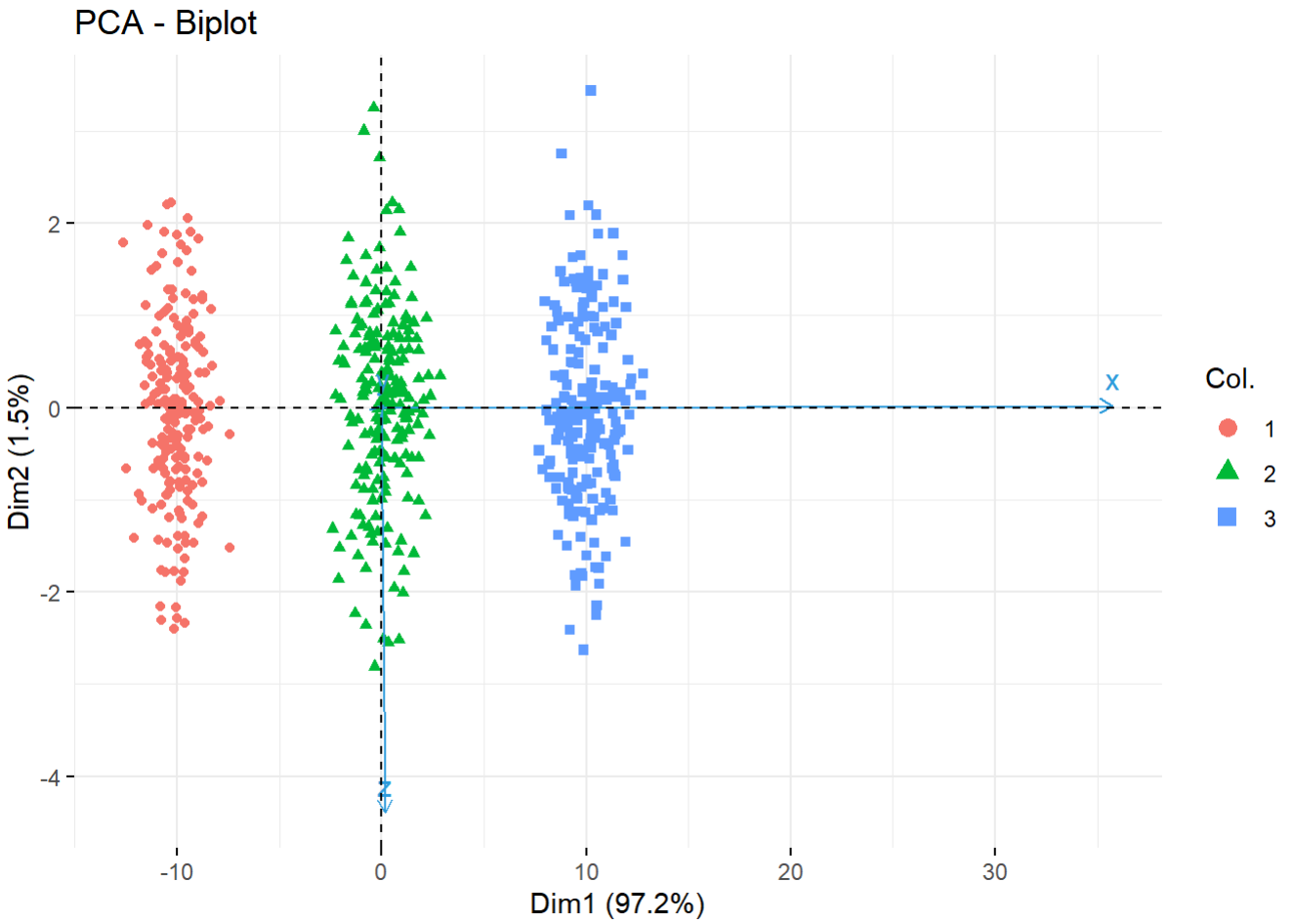
Now let us perform PCA.

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

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1  67.891405        97.242828                    97.24283
## Dim.2   1.024681         1.467680                    98.71051
## Dim.3   0.900276         1.289491                   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 possible to notice differences within the three groups.

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  1.94566 7.925126 6.038850
## 2 12.06630 8.002885 6.007017
## 3 21.96185 7.985679 6.140428
##
## Coefficients of linear discriminants:
##          LD1          LD2
## x 0.99827716 -0.002604012
## y 0.01822925 -0.544431058
## z 0.01279107  0.844750998
##
## Proportion of trace:
## LD1 LD2
##   1   0
```

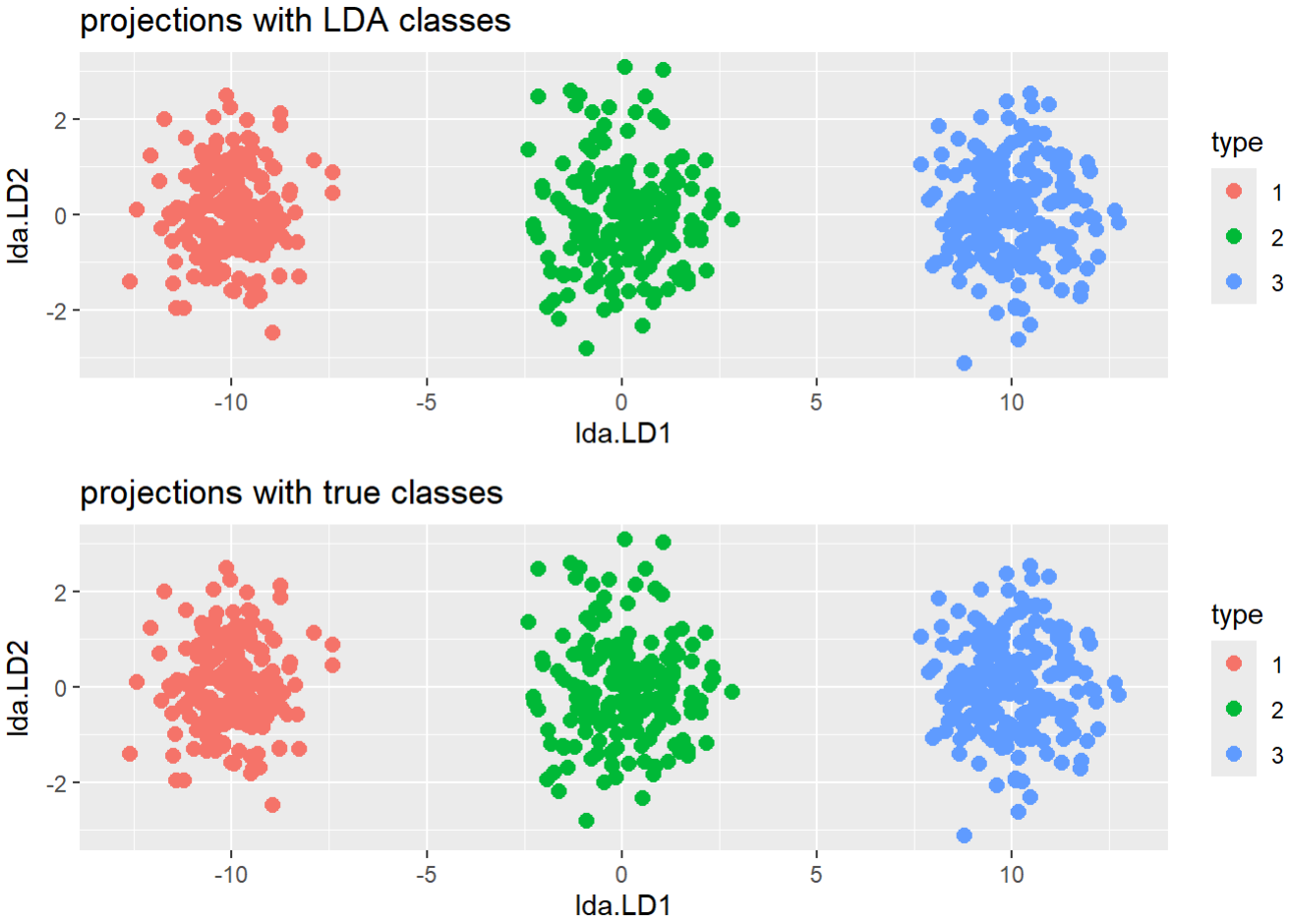
Let us plot the projections on LD1 and LD2

```
# 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

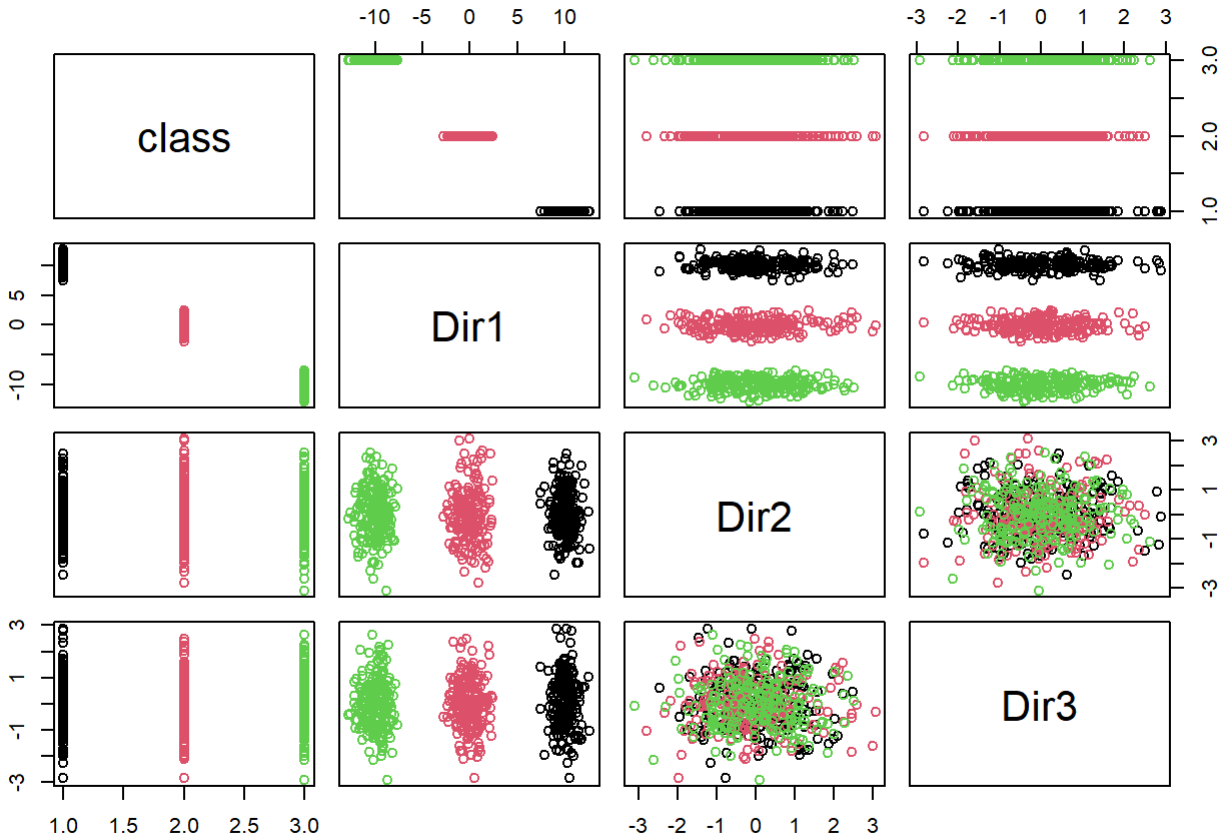
```
# default fitting method is "sir"
help(dr)

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 0.99975128 -0.002591068  0.005133845
## y 0.01825617 -0.541724804 -0.870013579
## z 0.01280996  0.840551916 -0.493001030
## Eigenvalues:
## [1]  9.852720e-01  2.051562e-03 -5.687426e-20
```

```
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"
```

SIR with continuous variable

We generated covariates in a classical continuous framework

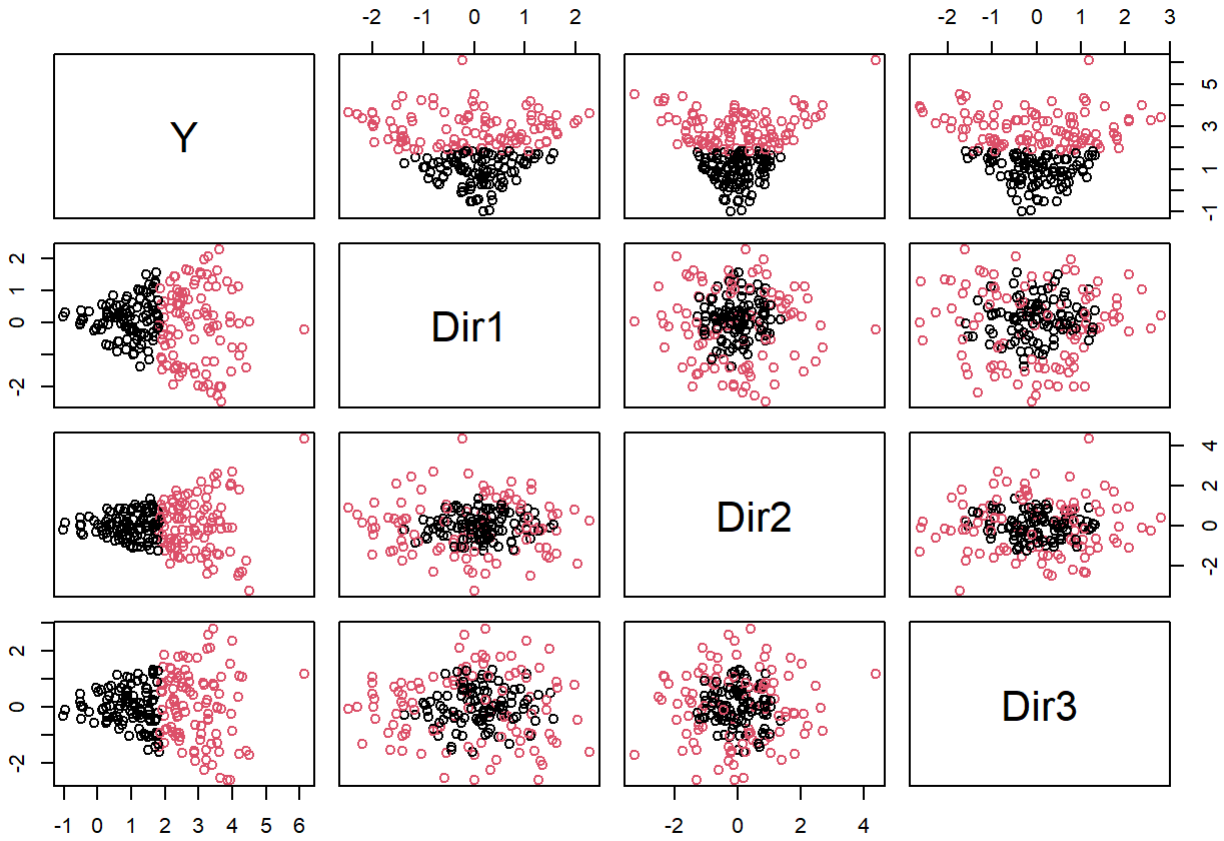
```
set.seed(1)
x <- mvrnorm(n, c(0,0,0), diag(3))
eps <- rnorm(n, 0, 0.3)
Y <- (x[,1]^2 + x[,2]^2 + x[,3]^2)^(1/2) +
  log((x[,1]^2 + x[,2]^2 + x[,3]^2)^(1/2)) + eps
df2 = data.frame(cbind(Y, x))

dr_res2 <- dr(Y ~ ., data = df2, method='sir', nslices=2)

summary(dr_res2)
```

```
##
## Call:
## dr(formula = Y ~ ., data = df2, method = "sir", nslices = 2)
##
## Method:
## sir with 2 slices, n = 200.
##
## Slice Sizes:
## 100 100
##
## Estimated Basis Vectors for Central Subspace:
##      V2      V3      V4
## -0.4236  0.3130  0.8501
##
##                Dir1
## Eigenvalues 0.01179
## R^2(OLS|dr) 0.57983
##
## Large-sample Marginal Dimension Tests:
##              Stat df p.value
## 0D vs >= 1D 2.357  3  0.5017
```

```
plot(dr_res2, col=dr_res2$slice.info$slice.indicator)
```



Now we perform the same analysis, but with 5 slices

```
Y <- (x[,1]^2 + x[,2]^2 + x[,3]^2)^(1/2) +
  log((x[,1]^2 + x[,2]^2 + x[,3]^2)^(1/2)) + eps
df2 = data.frame(cbind(Y, x))

dr_res2 <- dr(Y ~ ., data = df2, method='sir', nslices=5)

summary(dr_res2)
```

```
##
## Call:
## dr(formula = Y ~ ., data = df2, method = "sir", nslices = 5)
##
## Method:
## sir with 5 slices, n = 200.
##
## Slice Sizes:
## 40 40 40 40 40
##
## Estimated Basis Vectors for Central Subspace:
##       Dir1     Dir2     Dir3
## V2  0.3836 -0.89997 -0.04448
## V3  0.2032  0.03049  0.96656
## V4 -0.9009 -0.43489  0.25255
##
##                 Dir1     Dir2     Dir3
## Eigenvalues 0.01852 0.004285 4.86e-05
## R^2(OLS|dr) 0.87454 0.999623 1.00e+00
##
## Large-sample Marginal Dimension Tests:
##                  Stat df p.value
## 0D vs >= 1D 4.571726 12  0.9708
## 1D vs >= 2D 0.866749  6  0.9902
## 2D vs >= 3D 0.009721  2  0.9952
```

```
plot(dr_res2, col=dr_res2$slice.info$slice.indicator)
```