LDA

Pedro C. Silva

August 25, 2025

The main bibliographical reference for these notes is the section on Linear Discriminant Analysis included in the slides for the MMA course written by Professor J. Cadima, Introduction to Multivariate Analysis (2021-22). Following the same approach of Prof. Cadima, LDA is given here only as an exploratory analysis tool.

1 Notations and preliminary definitions

Let $\mathbf{X}_{N\times p} = [x_{ij}]$, $i = 1, \ldots, N$ and $j = 1, \ldots, p$, denote a dataset containing the observations of p (quantitative) variables across N individuals, with the set of individuals (rows) decomposed into k disjoint groups $C_1, C_2, \ldots, C_k$, with $n_1, n_2, \ldots, n_k$ individuals, respectively.

Each column (variable) j of $\mathbf{X}$ is denoted $\mathbf{x}_j = (x_{1j}, \ldots, x_{Nj})$ and each row (individual) i of $\mathbf{X}$ is denoted $\mathbf{x}^i = (x_{i1}, \ldots, x_{ip})$.

We denote by $\mathbf{X}^*$ the dataset of the centred cloud of individuals, that is, obtained by column centring $\mathbf{X}$. In other words each row i of $\mathbf{X}^*$ is equal to $\mathbf{x}^i - \mathbf{m}^G$, where

$$\mathbf{m}^G = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i,$$

is the centroid (mean vector) of the cloud of individuals. We can decomposed each row i of $\mathbf{X}^*$ as,

$$\mathbf{x}^i - \mathbf{m}^G = (\mathbf{x}^i - \mathbf{m}^\ell) \ + \ (\mathbf{m}^\ell - \mathbf{m}^G),$$

where,

$$\mathbf{m}^\ell = \frac{1}{n_\ell}\sum_{r\in C_\ell} \mathbf{x}^r,$$

denotes the centroid of the group of individuals that contains the individual (row) i. Applying this decomposition to all rows of $\mathbf{X}^*$ we end up with a matrix decomposition,

$$\mathbf{X}^* \ = \ \mathbf{X}_w \ + \ \mathbf{X}_b,$$

where the rows of $\mathbf{X}_w$ are the vectors $\mathbf{x}^i - \mathbf{m}^\ell$ (with ℓ depending on i) and the rows of $\mathbf{X}_b$ the vectors $\mathbf{m}^\ell - \mathbf{m}^G$. Note that:

- The matrix $\mathbf{X}_w$ is obtained from $\mathbf{X}^*$ by column centring each group of rows corresponding to the individuals belonging to a given class ℓ.
- The rows of $\mathbf{X}_b$ associated to each class ℓ consist of n_ℓ copies of $\mathbf{m}^\ell - \mathbf{m}^G$.

From the above matrix decomposition we obtain the following decomposition of the covariance matrix of $\mathbf{X}$,

$$\begin{aligned}\mathbf{S} = \text{cov}(\mathbf{X}) &= \frac{1}{N-1}(\mathbf{X}^*)^T\,\mathbf{X}^* = \frac{1}{N-1}\,(\mathbf{X}_w\ +\ \mathbf{X}_b)^T(\mathbf{X}_w\ +\ \mathbf{X}_b)\\ &= \frac{1}{N-1}\,(\mathbf{X}_w^T\ +\ \mathbf{X}_b^T)(\mathbf{X}_w\ +\ \mathbf{X}_b)\\ &= \frac{1}{N-1}(\mathbf{X}_w^T\,\mathbf{X}_w\ +\ \mathbf{X}_w^T\,\mathbf{X}_b\ +\ \mathbf{X}_b^T\,\mathbf{X}_w\ +\ \mathbf{X}_b^T\,\mathbf{X}_b).\end{aligned}$$

We can verify easily that $\mathbf{X}_w^T\,\mathbf{X}_b$ is the null matrix. Actually, each (i,j)-entry of this matrix has the form,

$$\sum_{\ell=1}^{k}(\mathbf{m}_i^\ell - \mathbf{m}_i^G)\sum_{r\in C_\ell}(\mathbf{x}_{rj} - \mathbf{m}_j^\ell),$$

where $\mathbf{m}_i^\ell$ and $\mathbf{m}_i^G$ are the i-th components of $\mathbf{m}^\ell$ and $\mathbf{m}^G$, resp., and $\sum_{r\in C_\ell}(\mathbf{x}_{rj} - \mathbf{m}_j^\ell)$ is equal to zero, since $\mathbf{m}_j^\ell$ is the mean of $\sum_{r\in C_\ell}\mathbf{x}_{rj}$. Hence $\mathbf{X}_b^T\,\mathbf{X}_w = (\mathbf{X}_w^T\,\mathbf{X}_b)^T$ is also the null matrix and therefore the decomposition of the covariance matrix of $\mathbf{X}$ above reduces to

$$\mathbf{S}\ =\ \frac{1}{N-1}(\mathbf{X}_w^T\,\mathbf{X}_w\ +\ \mathbf{X}_b^T\,\mathbf{X}_b),$$

that is,

$$\mathbf{S}\ =\ \mathbf{S}_w\ +\ \mathbf{S}_b.$$

Here:

- $\mathbf{S}_w\ =\ \frac{1}{N-1}\,\mathbf{X}_w^T\,\mathbf{X}_w\ =\ \text{cov}(\mathbf{X}_w)$ is the matrix of the within-class variability, which assess the dispersion of the individuals around their class centroids and equals,

$$\frac{1}{N-1}\sum_{\ell=1}^{k}\sum_{i\in C_\ell}(\mathbf{x}^i - \mathbf{m}^\ell)(\mathbf{x}^i - \mathbf{m}^\ell)^T.$$

- $\mathbf{S}_b\ =\ \frac{1}{N-1}\,\mathbf{X}_b^T\,\mathbf{X}_b\ =\ \text{cov}(\mathbf{X}_b)$ is the matrix of the between-class variability, which is a measure of the groups separability and equals,

$$\frac{1}{N-1}\sum_{\ell=1}^{k} n_\ell(\mathbf{m}^\ell - \mathbf{m}^G)(\mathbf{m}^\ell - \mathbf{m}^G)^T.$$

Note: we can alternatively define the above matrices in a slightly different manner as $\mathbf{S}_w\ =\ \mathbf{X}_w^T\,\mathbf{X}_w\ =\ (N-1)\text{cov}(\mathbf{X}_w)$ and $\mathbf{S}_b\ =\ \mathbf{X}_b^T\,\mathbf{X}_b\ =\ (N-1)\text{cov}(\mathbf{X}_b)$, which are then called within-class and between-class scatter matrices of $\mathbf{X}$. Both definitions lead to the same definition LDA...

We are going to use Fisher's iris flowers dataset with the language to illustrate the concepts and key ideas behind LDA.

The iris flower dataset can obviously be replaced by your own dataset with the set of individuals decomposed into disjoint classes.

[8]:
```
## raw data - X

# (non centred) iris dataset, excluding the categorical variable ''species''
X <- as.matrix(iris[,-5])

head(X) ## displays the 6 first rows of X
```

```
cls <- as.integer(iris[,5]) ## classes of the 150 iris flowers converted to integers
head(cls) ## displays the 6 first classes

lev <- unique(cls) ## distinct levels
lev ## displays the distinct levels

N <- nrow(X) # number of individuals
p <- ncol(X) # number of variables
k <- length(lev) # number of classes

N; p ; k ## returns these 3 numbers
```

A matrix: 6 × 4 of type dbl

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

1. 1 2. 1 3. 1 4. 1 5. 1 6. 1

1. 1 2. 2 3. 3

150

4

3

[10]:
```
## centred data - Xc

m.G <- colMeans(X) ### centroid of the individuals cloud (overall mean vector)
round(m.G,5) ## displays the centroid of X rounded to 6 digits

Xc <- scale(X, scale=FALSE) ## (column) centred data matrix

head(X-Xc) ## should be equal to 6 copies of m.G (since each row i of Xc = row i of X␣
↪- m.G)
```

Sepal.Length 5.84333 **Sepal.Width** 3.05733 **Petal.Length** 3.758 **Petal.Width** 1.19933

A matrix: 6 × 4 of type dbl

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.843333	3.057333	3.758	1.199333
5.843333	3.057333	3.758	1.199333
5.843333	3.057333	3.758	1.199333
5.843333	3.057333	3.758	1.199333
5.843333	3.057333	3.758	1.199333
5.843333	3.057333	3.758	1.199333

[12]:
```
# matrix of within-class variability: Sw

Xw=Xc # to start with

for (j in 1:k){  # j ranges from 1 to  k (number of groups)
```

```
  Xj <- Xc[cls==lev[j],] # submatrix of Xc containing the rows corresponding to␣
↪individuals of group j,
                        # i.e.,  with the same level lev[j].

  Xjc <- scale(Xj,scale=FALSE)  # Xjc = column centered submatrix  Xj.

  Xw[cls==lev[j],] <- Xjc  # replaces Xj by the column centred submatrix Xjc.
}

head(Xw) # displays the 6 first rows of Xw

Sw=cov(Xw)

Sw # displays the covariance matrix Sw
```

A matrix: 6 × 4 of type dbl

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
0.094	0.072	-0.062	-0.046
-0.106	-0.428	-0.062	-0.046
-0.306	-0.228	-0.162	-0.046
-0.406	-0.328	0.038	-0.046
-0.006	0.172	-0.062	-0.046
0.394	0.472	0.238	0.154

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.26145101	0.09147651	0.16526577	0.03788591
Sepal.Width	0.09147651	0.11383893	0.05450201	0.03227114
Petal.Length	0.16526577	0.05450201	0.18270201	0.04209262
Petal.Width	0.03788591	0.03227114	0.04209262	0.04131946

[14]:
```
# matrix of between-class variability: Sb

Xb = Xc - Xw

# displays 6 first rows of Xb = 6 copies of the centroid of the setosa group
head(Xb)

Sb=cov(Xb)

Sb # displays the covariance matrix Sb
```

A matrix: 6 × 4 of type dbl

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
-0.8373333	0.3706667	-2.296	-0.9533333
-0.8373333	0.3706667	-2.296	-0.9533333
-0.8373333	0.3706667	-2.296	-0.9533333
-0.8373333	0.3706667	-2.296	-0.9533333
-0.8373333	0.3706667	-2.296	-0.9533333
-0.8373333	0.3706667	-2.296	-0.9533333

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.4242425	-0.13391051	1.1090497	0.4783848
Sepal.Width	-0.1339105	0.07614049	-0.3841584	-0.1539105
Petal.Length	1.1090497	-0.38415839	2.9335758	1.2535168
Petal.Width	0.4783848	-0.15391051	1.2535168	0.5396868

[16]:
```
# overall covariance matrix: S

S <- cov(X) ##

S ; Sw + Sb # should be equal (S=Sb+Sw)
```

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.6856935	-0.0424340	1.2743154	0.5162707
Sepal.Width	-0.0424340	0.1899794	-0.3296564	-0.1216394
Petal.Length	1.2743154	-0.3296564	3.1162779	1.2956094
Petal.Width	0.5162707	-0.1216394	1.2956094	0.5810063

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.6856935	-0.0424340	1.2743154	0.5162707
Sepal.Width	-0.0424340	0.1899794	-0.3296564	-0.1216394
Petal.Length	1.2743154	-0.3296564	3.1162779	1.2956094
Petal.Width	0.5162707	-0.1216394	1.2956094	0.5810063

2 Review on PCA

We start by briefly reviewing the goal and definition of PCA, which has several points in common with LDA and some important differences that will be pointed out.

PCA aims to determine a projection of the cloud of the N observations of $\mathbf{X}_{N\times p}$ on a lower dimensional subspace such that the projection *preserves the variability of the dataset as much as possible*. This goal is accomplished by defining a new set of uncorrelated orthogonal variables, called principal components, that are linear combinations of the columns of $\mathbf{X}_{N\times p}$.

More precisely, the first principal component (PC1) is defined by the linear combinations $\mathbf{X}\,\mathbf{a}$, with $\|\mathbf{a}\| = 1$, maximizing the variance,

$$var(\mathbf{X}\mathbf{a}) = \mathbf{a}^T\mathbf{S}\,\mathbf{a},$$

which is equivalent to maximize the so-called Rayleight-ratio,

$$\frac{\mathbf{a}^T\mathbf{S}\,\mathbf{a}}{\mathbf{a}^T\,\mathbf{a}}, \qquad \text{with} \qquad \mathbf{a} \neq \vec{0}.$$

Here, $\mathbf{X}\mathbf{a}$ is a linear combination of the p columns $\mathbf{x}_1, ..., \mathbf{x}_p$ of $\mathbf{X}$, that is, $\mathbf{X}\mathbf{a} = \alpha_1\mathbf{x}_1 + \cdots \alpha_p\mathbf{x}_p$, with $\mathbf{a} = (\alpha_1, \ldots, \alpha_p)$.

It turns out that the (unit) vector of coefficients, also called loadings, $\mathbf{a}_1$ defining PC1 is an eigenvector of the covariance matrix $\mathbf{S}$ of $\mathbf{X}$ associated with its largest eigenvalue and the variance explained by PC1,

$$var(\mathbf{X}\mathbf{a}_1) = \mathbf{a}_1^T\,\mathbf{S}\,\mathbf{a}_1,$$

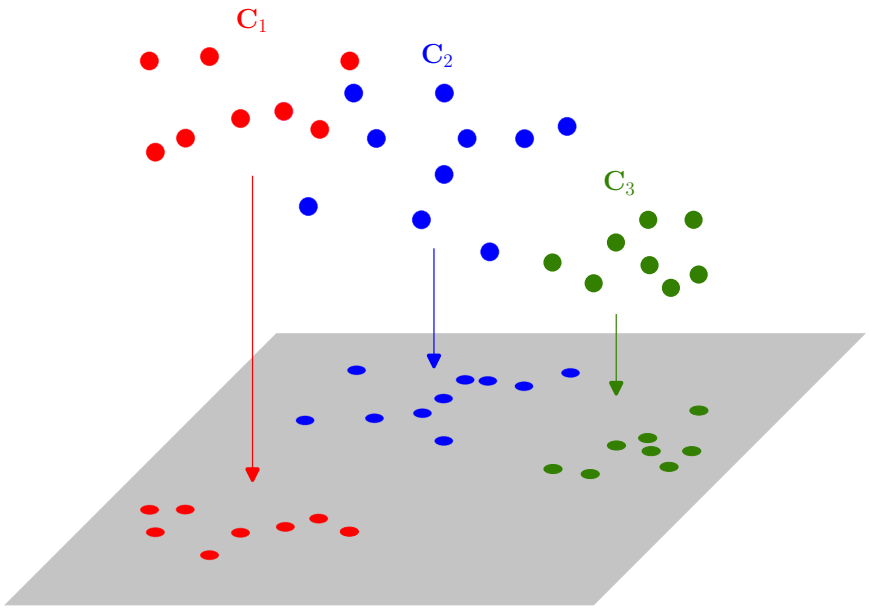
corresponds precisely to this eigenvalue λ_1.

The second principal component PC2 maximimizes the variance of the linear combinations $\mathbf{X}\mathbf{a}$ with $\|\mathbf{a}\| = 1$, subject to the additional constraint of $\mathbf{a}$ being orthogonal to $\mathbf{a}_1$. Similarly to the case of PC1, the vector of loadings, say $\mathbf{a}_2$, defininig PC2 is an eigenvector of $\mathbf{S}$ associated with the second largest eigenvalue and the variance explained by the linear combination $\mathbf{X}\mathbf{a}_2$, $\mathbf{a}_2^T\,\mathbf{S}\,\mathbf{a}_2$ is precisely the eigenvalue λ_2.

The process goes on along similar lines until we end up with p principal components PC1, PC2,..., PCp, defined by unit and pairwise orthogonal vectors, $\mathbf{a}_1$, $\mathbf{a}_2$,..., $\mathbf{a}_p$, associated to eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ of $\mathbf{S}$.

3 Motivation and goal of LDA

LDA aims to determine a projection of the cloud of the N observations defined by $\mathbf{X}_{N\times p}$ on a lower dimensional subspace that *optimally separates k groups of observations* $C_1 \ldots, C_k$ (see the figure below).



To accomplished this goal above LDA seeks a linear combination of the p columns of $\mathbf{X}^*$ that maximizes the ratio between-class variability over within-class variability, called Fisher discriminating ratio (FDR),that is, seeks to maximize the generalized Rayleight ratio,

$$\frac{var(\mathbf{X}_b\,\mathbf{a})}{var(\mathbf{X}_w\,\mathbf{a})} = \frac{\mathbf{a}^T\,\mathbf{S}_b\,\mathbf{a}}{\mathbf{a}^T\,\mathbf{S}_w\,\mathbf{a}}, \qquad \text{with} \qquad \mathbf{a} \neq \vec{0},$$

where $\mathbf{S}_b$ and $\mathbf{S}_w$ are respectively, the matrices of the between-class and within-class variabilities previously defined.

The linear combination $\mathbf{X}^*\mathbf{a}_1$ maximizing the above ratio is called the first discriminant function or discriminant axis (LDA1) and the value of the ratio is the corresponding axis discriminating capacity.

It can be proved (see the **Appendix**) that the first discriminant axis (LDA1) is a linear combinations of the of centred variables with coefficients, also called loadings, given by an eigenvector $\mathbf{a}_1$ of the matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$ associate with its largest eigenvalue λ_1, with the discriminating capacity of LDA1 being precisely equal to λ_1.

The second discriminant axis (LDA2) is defined by a linear combination $\mathbf{X}^*\mathbf{a}_2$ that maximizes the same ratio

$$\max \frac{\mathbf{a}^T\mathbf{S}_b\,\mathbf{a}}{\mathbf{a}^T\mathbf{S}_w\,\mathbf{a}}, \qquad \text{with} \qquad \mathbf{a} \neq \vec{0},$$

subjet to the additional constraint of being **uncorrelated** with LDA1. It can be proved that $\mathbf{a}_2$ is an eigenvector associated with the second largest eigenvalue λ_2 of matrix $\mathbf{S}_w^{-1}S_b$ with ratio (discriminating capacity) given precisely by λ_2.

This process continues until we reach the smallest nonzero eigenvalue of $\mathbf{S}_w^{-1}\mathbf{S}_b$. The number of nonzero eigenvalues is, at most, $k-1$, which implies that the number of discriminant axes cannot exceed the number of classes minus one, even when the number of variables is large.

The previous result requires the invertibility of $\mathbf{S}_w$, which can only occur when $k < N - p$.

Historical note (TBD) Relation with FDR of the original paper by R. Fisher (1936), "The Use of Multiple Measurements in Taxonomic Problems". Annals of Eugenics. 7 (2): 179–188, containing the pioneering ideas of LDA.

4 Eigendata of $\mathbf{S}_w^{-1}\mathbf{S}_b$

[25]:
```
invSw <- solve(Sw) # solve(Sw) = inverse matrix of Sw
eigenData<-eigen(invSw %*% Sb)
eigenData
```

```
eigen() decomposition
$values
[1]  3.219193e+01  2.853910e-01  5.908344e-14 -4.743822e-14

$vectors
           [,1]         [,2]        [,3]       [,4]
[1,] -0.2087418 -0.006531964  0.02707283 -0.6735434
[2,] -0.3862037 -0.586610553 -0.34535300  0.4399771
[3,]  0.5540117  0.252561540 -0.41495379  0.4683326
[4,]  0.7073504 -0.769453092  0.84131547 -0.3652725
```

Since in the iris case we have 3 groups, we can have, at most, 2 nonzero eigenvalues, and thus 2 discriminant axes. Actually,

[28]:
```
round(eigenData$values,5)
```

1. 32.19193 2. 0.28539 3. 0 4. 0

The eigenvectors (vectors of loadings) and respective eigenvalues of $\mathbf{S}_w^{-1}\mathbf{S}_b$ defining the discriminant axes and corresponding discriminating capacities are always real vectors and real numbers, resp., even though the square matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$ is no longer symmetric.

However these eigenvectors and eigenvalues often appear with zero imaginary parts and thus have to be converted to real vectors/values using the function `Re()`, which returns the real part of a complex vector or number...

The vectors of loadings containing the coefficients of each discriminant axis w.r.t. the original variables, can be retrieved from the `eigenData` as follows:

[32]:
```
loadings.LDA<-Re(eigenData$vectors[,1:(k-1)])
loadings.LDA
```

A matrix: 4 × 2 of type dbl

-0.2087418	-0.006531964
-0.3862037	-0.586610553
0.5540117	0.252561540
0.7073504	-0.769453092

Note that in general, and contraly to PCA, the matrix of loadings say $\mathbf{V}$, is no longer orthogonal, that is, the $\mathbf{V}^T\mathbf{V}$ is no longer the identity matrix.

This can be verified running the code below where the loadings matrix $\mathbf{V}$ is denoted `loadings.LDA`:

[35]:
```
t(loadings.LDA) %*% loadings.LDA
```

A matrix: 2 × 2 of type dbl

1.0000000	-0.1764362
-0.1764362	1.0000000

The discriminating capacities for all discriminant axes can be retrieved from the `eigenData` as follows:

[38]:
```
discrCapacities<-Re(eigenData$values[1:(k-1)])
round(discrCapacities,5)
```

1. 32.19193 2. 0.28539

We denote by λ_1 and λ_2, respectively, the largest and second largest eigenvalue of $\mathbf{S}_w^{-1}\mathbf{S}_b$:

[41]:
```
lambda1=discrCapacities[1]
round(lambda1,5)

lambda2=discrCapacities[2]
round(lambda2,5)
```

32.19193

0.28539

The first discriminant axis (LDA1) is the linear combination of the centred variables SL, SW, PL and PW,

$$\mathbf{X}\,\mathbf{a}_1 = -0.2087418\,SL - 0.3862037\,SW + 0.5540117\,PL + 0.7073504\,PW,$$

where $\mathbf{a}_1 = (-0.2087418, -0.3862037, 0.5540117, 0.7073504)$ is the eigenvector associated with the largest eigenvalue $\lambda_1 = 32.19193$ of $\mathbf{S}_w^{-1}\mathbf{S}_b$, that is, to the first column of the loadings matrix. The two variables associated with petal measurements, and particularly the `Petal Width` variable, have greater absolute values of the corresponding loading coefficients and thus are more important to discriminate the iris species.

The discriminating capacity λ_1 of LDA1 is approximately equal to 32.192, meaning that the *between-class variability of iris flowers projected on the first discriminant axis LDA1 is about 32 times the corresponding within-class variability*, that is,

$$\frac{\mathbf{a}_1^T\,\mathbf{S}_b\,\mathbf{a}_1}{\mathbf{a}_1^T\,\mathbf{S}_w\,\mathbf{a}_1} = \lambda_1 \approx 32.$$

The following code confirms this.

[45]:
```
a1 <- loadings.LDA[,1] ### second column of the loadings matrix
a1
round(lambda1,5)
t(a1)%*% Sb %*% a1 / (t(a1)%*% Sw %*% a1) ;  # should be equal to lambda1
```

1. -0.20874182147455 2. -0.386203686755055 3. 0.554011715552863 4. 0.707350396433383

32.19193

A matrix: 1 × 1 of type dbl 32.19193

[47]:
```
a1 <- loadings.LDA[,1] ### second column of the loadings matrix
a1
round(lambda1,5)
t(a1)%*% Sb %*% a1 / (t(a1)%*% Sw %*% a1) ;  # should be equal to lambda1
```

1. -0.20874182147455 2. -0.386203686755055 3. 0.554011715552863 4. 0.707350396433383

32.19193

A matrix: 1 × 1 of type dbl 32.19193

The discriminating capacity λ_2 of LDA2 is approximately 0.285 (the 2nd largest eigenvalue of the matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$), which means that *between-class variability of the 150 iris flowers projected on the second discriminant axis is about one fourth of the corresponding within-class variability.*

Actually, denoting by $\mathbf{a}_2$ the eigenvector associated with the 2nd largest eigenvalue (second column of the loadings.LDA matrix), we obtain,

$$\frac{\mathbf{a}_2^T \mathbf{S}_b \mathbf{a}_2}{\mathbf{a}_1^T \mathbf{S}_w \mathbf{a}_2} = \lambda_2 \approx 0.285,$$

as we can check executing the code below.

[50]:
```
a2 <- loadings.LDA[,2] ### second column of the loadings matrix
a2
round(lambda2,5)
t(a2)%*% Sb %*% a2 / (t(a2)%*% Sw %*% a2) ;  # should be equal to lambda2
```

1. -0.00653196404724265 2. -0.586610553124505 3. 0.252561540044391 4. -0.76945309207193

0.28539

A matrix: 1 × 1 of type dbl 0.285391

The sum of all discriminating capacities $\lambda_1 + \lambda_2$, denoted J (or sometimes J_3), represents the discriminatory information of $\mathbf{X}$ that we want to maximize and equals the sum of the eigenvalues of $\mathbf{S}_w^{-1}\mathbf{S}_b$, which in turn is equal to the trace of $\mathbf{S}_w^{-1}\mathbf{S}_b$.

In other words,

$$J = \text{tr}(\mathbf{S}_w^{-1}\mathbf{S}_b) = \lambda_1 + \lambda_2.$$

The proportions of the trace accounted by each one of the discriminant axis LDA1 and LDA2, that is,

$$\frac{\lambda_1}{J}, \qquad \frac{\lambda_2}{J},$$

give the relative importance of each axis to recover the discriminatory information of X and play a role similar to the role of the proportions of explained variance by each principal component in PCA.

In iris flowers case we can compute the trace J and the two proportions $\frac{\lambda_1}{J}$ and $\frac{\lambda_2}{J}$ as follows.

[53]:
```
## proportion of trace

J=sum(diag( invSw %*% Sb)) ## trace of the matrix Sw^(-1)Sb
round(J,5)  ### discriminant power rounded to 5 digits
round(lambda1/J,5) ### prop. of discriminating capacity accounted by LDA1
round(lambda2/J,5) ### prop. of discriminating capacity accounted by LDA1
```

32.47732

0.99121

0.00879

LDA1 alone is responsible for recovering more than 99% of the discriminatory information and thus suffices to discriminate the 3 iris species! This can also be observed by plotting the iris flowers in the plane defined by the 2 discriminant axes LDA1 and LDA2. But in order to that we first need to obtain the coordinates of iris flowers w.r.t the discriminant axes, called discriminant scores.

Similarly to PCA, the matrix of discriminant scores containing the coordinates of the iris flowers for all discriminant axes, is given by the product of the centred data matrix $\mathbf{X}^*$ by the loadings matrix $\mathbf{V}$ (denoted `loadings.LDA` in the code):

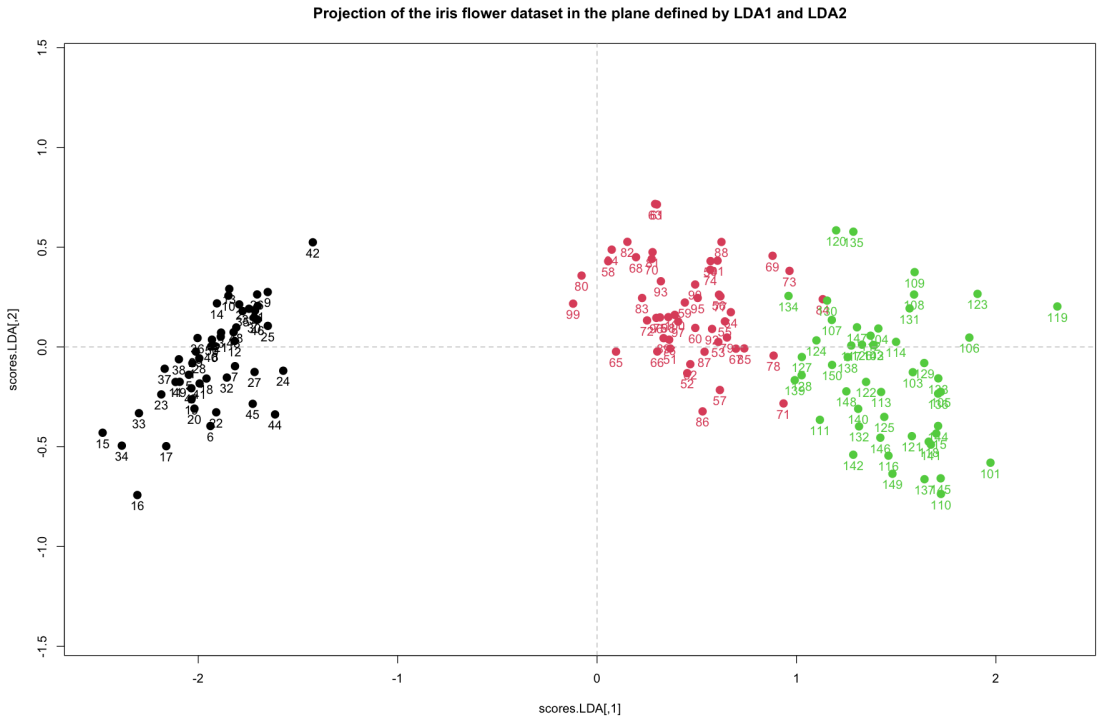
[57]:
```
scores.LDA <- Xc %*% loadings.LDA
head(scores.LDA)
```

A matrix: 6 × 2 of type dbl

-2.029033	-0.0814175
-1.794183	0.2131942
-1.885077	0.0719223
-1.714780	0.1817489
-2.046779	-0.1394254
-1.938464	-0.3961435

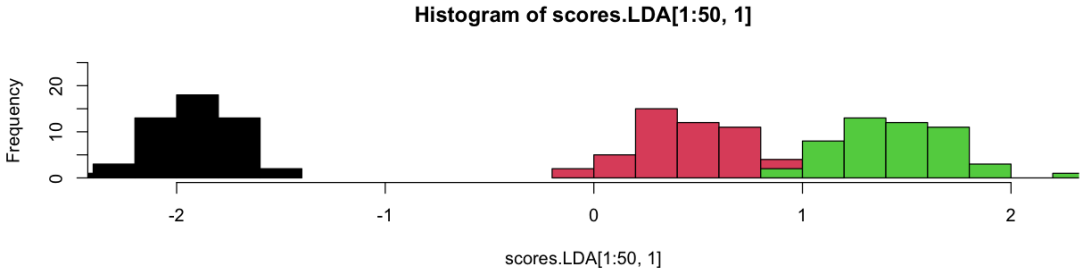
The projected 150 iris flowers in the plane defined by LDA1 and LDA2 can now be plotted using the discriminant scores as coordinates, with each dot colored and labeled according to the class of the respective iris flower:

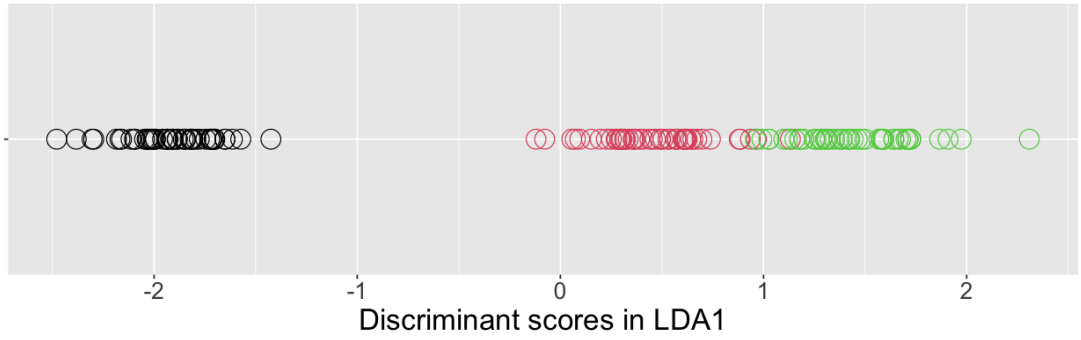
[60]:
```
library(repr)
par(mfrow=c(1,1))

options(repr.plot.width =15, repr.plot.height = 10) ## plot windows dimensions

plot(scores.LDA,col=cls,pch=16,asp=TRUE,cex=1.5,main="Projection of the iris flower␣
 ↪dataset in the plane defined by LDA1 and LDA2")

abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")

text(scores.LDA[,1],scores.LDA[,2],labels=1:N, col=cls,cex=1,pos=1)
```



It is clear from the previous plot that essentially all separation of the iris species is accomplished using only the first discriminant axis, as it would be expected from the proportions of trace on LDA1 (>99%) and LDA2 (<1%). The following projections on LDA1 and LDA2 axes together with the histograms of the distributions of the discriminant scores of the 3 species in LDA1 and LDA2 reinforce this idea.

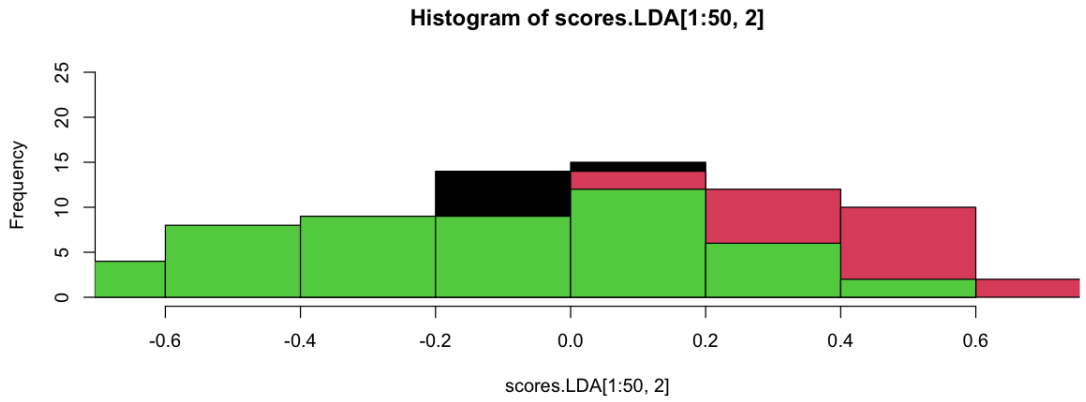
[63]:
```
### PROJECTION AND HISTOGRAM ON LDA1

library(ggplot2)
library(repr)

df_1 = data.frame(x = scores.LDA[,1], y = rep(0,N))

options(repr.plot.width = 10.5, repr.plot.height = 3)

hist(scores.LDA[1:50,1],xlim=c(-2.25,2.15),ylim=c(0,25),col=cls[1:50])
hist(scores.LDA[51:100,1],col=cls[51:100],add=TRUE)
hist(scores.LDA[101:150,1],col=cls[101:150],add=TRUE)

options(repr.plot.width = 10.65, repr.plot.height = 4)

ggplot(data=df_1, aes(x = x, y = "")) +
    geom_point(size = 6,color=cls,pch=1) +
    theme(text = element_text(size = 20), element_line(linewidth = 10)) +
    labs(title="", subtitle = "",
       x="Discriminant scores in LDA1",
       y="")
```

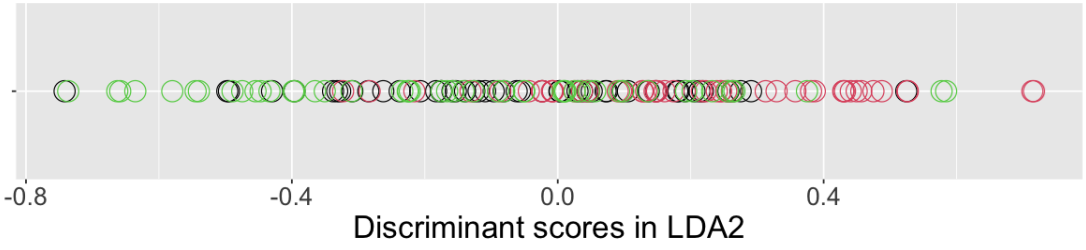




With regard to the second discriminant axis LDA2, the projection of the 3 iris classes appear rather mixed.

[65]:
```
### PROJECTION AND HISTOGRAM ON LDA2

library(ggplot2)
library(repr)

df_2 = data.frame(x = rep(0,N), y = scores.LDA[,2])


options(repr.plot.width = 10, repr.plot.height = 4)

hist(scores.LDA[1:50,2],xlim=c(-0.65,0.7),ylim=c(0,25),col=cls[1:50])
hist(scores.LDA[51:100,2],col=cls[51:100],add=TRUE)
hist(scores.LDA[101:150,2],col=cls[101:150],add=TRUE)

options(repr.plot.width = 10, repr.plot.height = 3)

ggplot(data=df_2, aes(x = y, y = "")) +
    geom_point(size = 6,color=cls,pch=1) +
    theme(text = element_text(size = 20), element_line(linewidth = 10)) +
    labs(title="", subtitle = "",
       y="",
       x="Discriminant scores in LDA2")
```





5 LDA vs PCA

To stress the idea that LDA and PCA have different purposes, we are going to assess the discriminating capacities of PCA and the explained variances of LDA, in the case of iris flowers dataset.

To compute the discriminating capacities of PC1 and PC2 we just have to compute the ratios

$$\frac{\mathbf{u}_1^T \mathbf{S}_b \mathbf{u}_1}{\mathbf{u}_1^T \mathbf{S}_w \mathbf{u}_1} \quad \text{and} \quad \frac{\mathbf{u}_2^T \mathbf{S}_b \mathbf{u}_2}{\mathbf{u}_2^T \mathbf{S}_w \mathbf{u}_2},$$

where $\mathbf{u}_1$ and $\mathbf{u}_2$ are the vectors of loadings for PC1 and PC2, respectively.

[69]:
```
X.ACP <-  prcomp(X) ## performs PCA on the covariance matrix of X
X.ACP
```

```
Standard deviations (1, .., p=4):
[1] 2.0562689 0.4926162 0.2796596 0.1543862

Rotation (n x k) = (4 x 4):
                    PC1         PC2         PC3        PC4
Sepal.Length  0.36138659 -0.65658877  0.58202985  0.3154872
Sepal.Width  -0.08452251 -0.73016143 -0.59791083 -0.3197231
Petal.Length  0.85667061  0.17337266 -0.07623608 -0.4798390
Petal.Width   0.35828920  0.07548102 -0.54583143  0.7536574
```

The discriminating capacity of PCA1,

$$\frac{\mathbf{u}_1^T \mathbf{S}_b \mathbf{u}_1}{\mathbf{u}_1^T \mathbf{S}_w \mathbf{u}_1},$$

equals (approx.) 13.242 and thus the first principal component as good discriminating capacity, although considerably inferior to the discriminating capacity of LDA1 (32.19193), as it was somehow expected, since PCA maximizes the explained variances but not the groups discrimination.

[72]:
```
### discriminating capacity of LDA1

loadings.ACP<-X.ACP$rotation

u1<-loadings.ACP[,1] # vector of coefficients defining the first PC
round(u1,5)

discr.acp.1<-(t(u1) %*% Sb %*% u1)/(t(u1) %*% Sw %*% u1)
discr.acp.1
```

Sepal.Length 0.36139 **Sepal.Width** -0.08452 **Petal.Length** 0.85667 **Petal.Width** 0.35829

A matrix: 1 × 1 of type dbl 13.24182

Likewise, the discriminating capacity of PCA2 is the ratio computed below and equals (approx.) 0.160, also inferior to the discriminating capacity of LDA2 (0.28539).

[75]:
```
### discriminating capacity of LDA1

u2<-loadings.ACP[,2] # vector of coefficients defining the second PC
round(u2,5)

discr.acp.2<-(t(u2) %*% Sb %*% u2)/(t(u2) %*% Sw %*% u2)
discr.acp.2
```

Sepal.Length -0.65659 **Sepal.Width** -0.73016 **Petal.Length** 0.17337 **Petal.Width** 0.07548

A matrix: 1 × 1 of type dbl 0.1599517

The explained variances of PC1, PC2, PC3 and PC4 can be obtained as follows:

[77]:
```
round(X.ACP$sdev^2,5)
```

1. 4.22824 2. 0.24267 3. 0.07821 4. 0.02384

which are also equal to the variances of PCA scores,

[80]:
```
round(var(X.ACP$x[,1]),5); round(var(X.ACP$x[,2]),5); round(var(X.ACP$x[,3]),5);␣
↪round(var(X.ACP$x[,4]),5)
```

4.22824

0.24267

0.07821

0.02384

We can also retrieve these variances from the diagonal elements of the covariance the PCA scores matrix:

[83]:
```
round(diag(var(X.ACP$x)),5)
```

PC1 4.22824 **PC2** 0.24267 **PC3** 0.07821 **PC4** 0.02384

To compute the explained variances of LDA1 and LDA2 we proceed in a similar fashion w.r.t. the discriminant scores:

[86]:
```
round(diag(var(scores.LDA)),5)
```

1. 2.07433 2. 0.09314

As expected, they are considerable inferior to the variances explained by the corresponding principal components, as it was expected since LDA maximizes the groups discrimination but not the explained variances!

Some remarks The vectors of loadings $\mathbf{u}_1$ and $\mathbf{u}_2$ defining the principal components PC1 and PC2 are always pairwise orthogonal, i.e., $\mathbf{u}_2^T\mathbf{u}_1 = 0$, which implies, in particular, that the linear combinations $\mathbf{X}\mathbf{u}_1$ and $\mathbf{X}\mathbf{u}_2$ defining PC1 and PC2 are noncorrelated, that is, $cov(\mathbf{X}\mathbf{u}_1, \mathbf{X}\mathbf{u}_2) = 0$. We verify these properties running the code below:

[91]:
```
u1 ; u2

round( t(u2) %*% u1, 7)
# u1^T u1 is zero  =>  u1 and u2 are pairwise orthogonal

round( cov(X %*% u1, X %*% u2), 7)
## cov(X u1,X u2) = 0 => PC1 and PC2 are noncorrelated
```

Sepal.Length 0.361386591785368 **Sepal.Width** -0.0845225140645685 **Petal.Length** 0.856670605949835 **Petal.Width** 0.35828919715155

Sepal.Length -0.656588771286842 **Sepal.Width** -0.730161434785027 **Petal.Length** 0.173372662795857 **Petal.Width** 0.0754810199174639

A matrix: 1 × 1 of type dbl 0

A matrix: 1 × 1 of type dbl 0

Contrarily to PCA, the vectors of loadings $\mathbf{a}_2$ and $\mathbf{a}_1$ defining the discriminant axes LDA1 and LDA2 are no longer orthogonal, i.e., $\mathbf{a}_2^T \mathbf{a}_1 \neq 0$ (in general), but only pairwise $\mathbf{S}_w$-orthogonal:

$$\mathbf{a}_2^T \, \mathbf{S}_w \, \mathbf{a}_1 \;=\; 0.$$

Nevertheless the linear combinations $\mathbf{X}^*\mathbf{a}_1$ and $\mathbf{X}^*\mathbf{a}_2$ are still noncorrelated, i.e, $cov(\mathbf{X}^*\mathbf{a}_1, \mathbf{X}^*\mathbf{a}_2) = 0$, which means that the discriminant LDA1 and LDA2 axes are noncorrelated. We can verify the above properties with the following code:

[94]:
```
a1 ; a2

round( t(a2) %*% a1, 7)
# a_2^Ta_1 is not zero  =>  a1 and a2 are not pairwise orthogonal

round( t(a2) %*% Sw %*% a1, 7)
# a_2^T Sw a_1 = 0 => a1 and a2 are pairwise Sw-orthogonal
round( cov(Xc %*% a1, Xc %*% a2), 7)
## cov(Xc a_1,Xc a_2) = 0 => LDA1 and LDA2 are noncorrelated
```

1. -0.20874182147455 2. -0.386203686755055 3. 0.554011715552863 4. 0.707350396433383

1. -0.00653196404724265 2. -0.586610553124505 3. 0.252561540044391 4. -0.76945309207193

A matrix: 1 × 1 of type dbl -0.1764362

A matrix: 1 × 1 of type dbl 0

A matrix: 1 × 1 of type dbl 0

6 How to predict the class of a new element?

We want to use Fisher's LDA to predict the class of a new vector $\mathbf{y} \in \mathbb{R}^p$ containing measurements of some iris flower of an unknown species. We can predict the species of $\mathbf{y}$ using only one or both discriminant axes. But as we have seen before, in the case of iris dataset all discrimination is essencially achieved by the first discriminant axis and thus one axis will suffice.

The idea is to determine the coordinate of $\mathbf{y}$ projected on LDA1, called (first) discriminant score, denoted $L_1(\mathbf{y})$, and the centroids $\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_k$ of the k groups of the discriminant scores arising from the k original groups (which is equivalent to compute the scores of the centroids of the original groups) and assign to $\mathbf{y}$ the class j whose centroid $\bar{\mathbf{z}}_j$ is the closest to the score $L_1(\mathbf{y})$.

In other words we assign to $\mathbf{y}$ to group j s.t.

$$|L_1(\mathbf{y}) - \bar{\mathbf{z}}_j)| \leq |L_1(\mathbf{y}) - \bar{\mathbf{z}}_i|, \qquad \forall i = 1, \ldots, k,$$

which is equivalently to

$$j = \text{argmin}_{1 \leq i \leq k} \; |L_1(\mathbf{y}) - \bar{\mathbf{z}}_i|.$$

The above definition does not account for the distribution of the scores of each group in LDA1. For instance, if $L_1(\mathbf{y})$ is equidistant to the centroids $\bar{z}_i$ and $\bar{z}_j$, it is more likely that $\mathbf{y}$ belongs to the group having a greater dispersion and the classification criterion above should account for this. In order to incorporate the distribution of the discriminant scores in the classification of a new element $\mathbf{y}$, we can divide the distance between $L_1(y)$ and each centroid $\bar{\mathbf{z}}_j$ by the standard deviation s_j of the discriminant scores belonging to class j.

In other words, we assign to $\mathbf{y}$ the group j s.t.

$$\frac{|L_1(\mathbf{y}) - \bar{\mathbf{z}}_j)|}{s_j} \leq \frac{|L_1(\mathbf{y}) - \bar{z}_i|}{s_i}, \qquad \forall i = 1, \ldots, k.$$

When multiple discriminant axes are used to predict the class of a new element, the normalization of the distances by the standard deviations above can be replaced by the Mahalanobis distances...

The first discriminant score of an arbitrary $\mathbf{y} \in \mathbb{R}^p$ can be computed as follows:

$$L_1(\mathbf{y}) = \mathbf{y}^T \mathbf{a}_1,$$

where $\mathbf{a}_1$ is the vector of loadings defining LDA1, that is, the first column of the loadings matrix.

Likewise, the j-discriminant score of $\mathbf{y}$, $L_j(\mathbf{y})$, $1 \leq j \leq k-1$, can be computed as,

$$L_j(\mathbf{y}) = \mathbf{y}^T \mathbf{a}_j,$$

where $\mathbf{a}_j$ is the vector of loadings defining LDAj, i.e., the j column of the loadings matrix.

Thus, the set of $k-1$ discriminant scores of $\mathbf{y}$, denoted $L(\mathbf{y})$, are computed at once, as

$$L(\mathbf{y}) = \mathbf{y}^T \mathbf{V},$$

where $\mathbf{V}$ is the loading matrix (denoted `loadings.LDA` in the code). We call $L(\mathbf{y})$, Fisher's discriminant function.

Note: we are assuming here that $\mathbf{y}$ contains already the coordinates with respect to the centroid of the (training data) $\mathbf{m}^G$ (or that $\mathbf{X}$ is already centred). Otherwise, we need in the first place to replace $\mathbf{y}$ by $\mathbf{y} - \mathbf{m}^G$.

Let us apply the previous ideas and concepts to classify a new iris flower with vector of hypothetical sepal and petal measurements $\mathbf{y} = (6.2, 3.2, 5, 1.9)$. But, as pointed out, we first need to replace $\mathbf{y}$ by the vector of coordinates w.r.t the centroid $\mathbf{m}^G$, $\mathbf{y} - \mathbf{m}^G$, that we still denote by $\mathbf{y}$:

[103]:
```
### new unknwon flower that we pretend to classify

y <- c(6.2, 3.2, 5, 1.9) ## original sepal and petal measurements of an hypothetical␣
 ↪iris flower of unknown species

m.G <- as.vector(colMeans(X)) ## centroid of the training data (iris dataset)

y <- y - m.G # replaces y by y - m.G (coordinates of y w.r.t the centroid m.G)

dim(y) <- c(4,1)  # ensures that y is a column matrix of type p x 1

y
```

A matrix: 4×1 of type dbl

0.3566667
0.1426667
1.2420000
0.7006667

The discriminant score of the new flower on LDA1 can now be computed as $L_1(\mathbf{y}) = \mathbf{y}^T\mathbf{V}$, where $\mathbf{V}$ denotes the loadings matrix (denoted `loadings.LDA` in the code):

[106]:
```
scores.y <- t(y) %*% loadings.LDA  ## L(y) = (L1(y),L2(y))

scores.y[1] ### L1(y)
```

```
scores.y[2] ### L2(y)
```

1.054149752848

-0.311468206532545

Recall that for each $j = 1, \ldots, k-1$, the j column of the scores matrix (called `scores.LDA` in the code) contains the discriminant scores on the j discriminant axis of the individuals (rows) of the (column centred) data matrix $\mathbf{X}$. In particular, the first column of the scores matrix contains discriminant scores on LDA1 of the (centred) cloud of individuals.

Hence the centroids $\bar{\mathbf{z}}_j$, $j = 1, \ldots, k$ of the $k = 3$ groups of discriminant scores on LDA1 corresponding to the 3 species of iris flowers, setosa, versicolor and virginica, can be easily computed as follows.

[109]:
```
bar.z <- rep(NA,k) ## empty vector with k components
for (i in 1: k){
  bar.z [i]<-mean(scores.LDA[cls==lev[i],1])
}
bar.z ## centroids of the scores (in LDA1) of setosa, versicolor and virginica flowers.
```

1. -1.91471795821838 2. 0.459337381960075 3. 1.4553805762583

Now we compute the absolute values of the differences between $L_1(\mathbf{y})$ and each one of the centroids $\bar{\mathbf{z}}_j$, $j = 1, \ldots, k$, of the k groups of discriminant scores on LDA1:

[112]:
```
## rep(scores.y[1],k): vector containing k copies of L1(y)
d.cls=abs((rep(scores.y[1],k) - bar.z))
d.cls
```

1. 2.96886771106638 2. 0.594812370887925 3. 0.401230823410304

In order to determine the closest centroid we first determine the permutation of the indices $j = 1, \ldots, k$, that sorts the previous differences by increasing order and then we pick the first indice:

[115]:
```
ord <- order(d.cls)
### gives the order of the indices that increasingly
### sort the values of the vector d.cls

ord
ord[1]
```

1. 3 2. 2 3. 1

3

The predicted class corresponding to the closest centroid can be retrieved as the level of the vector `lev` associated with the first indice of the permutation vector `ord`, i.e., with `ord[1]`:

[118]:
```
pred.y<-lev[ord[1]]
pred.y
```

3

7 Construction of a classifier based on Fisher LDA

In order to be more easy the replicate the above steps on an arbitrary data frame containing some training data that defines the discriminant functions (axes) and an arbitrary data frame containing some testing data that we pretend to classify, we are going to construct a classifier function named L.discr(), which

applies the ideas and concepts explained above and uses all possible discriminant axes to predict the classes of the testing data.

[121]:
```
L.discr <- function(df.train,cls.train,df.test=df.train){

    # df.train: a numerical dataframe or data.matrix (Nxp) used as training data

    # cls.train: a vector of length N containing the levels of each individual (row).

    # df.test: a numerical (Mxp) dataframe or data.matrix, whose rows
    # we pretend classify. By default df.test = df.train, if omitted.


    X.train <- as.matrix(df.train)  # converts data.frame to a data matrix

    X.test <- as.matrix(df.test)  # converts data.frame to a data matrix

    p = ncol(X.train)  # number of variables

    N = nrow(X.train)  # number of training individuals, already classified

    M = length(X.test)/p  # number of testing individuals, i,e., to be classified

    dim(X.test) <- c(M,p) # to ensure  X.test is defined as a M x p matrix

    Xc <- scale(X.train,scale=FALSE)  # Xc = column centred training data

    m.G <- colMeans(X.train) # centre of gravity of the training data

    X.test.c <- X.test-matrix(rep(m.G,M),nrow=M,byrow=TRUE)
    # computes the coordinates of the testing individuals
    # w.r.t. the centroid m.G of the training data

    cls <- as.vector(cls.train)  # to ensure cls is defined as a vector

    lev <- unique(cls)  # vector containing the distinct (factor) levels

    k = length(lev)  # number of distinct (factor) levels

    S = cov(X)  ## overall covariance matrix


    ### WITHIN-CLASS AND BETWEEN-CLASS SCATTER MATRICES Sw AND Sb

    Xw = Xc

    for (j in 1:k){
        Xj <- Xc[cls==lev[j],]
        Xw[cls==lev[j],] <- scale(Xj,scale=FALSE)
    }

    Sw = cov(Xw)

    Xb = Xc-Xw
```

```r
Sb = cov(Xb) # should be equal to S-Sw

invSw = solve(Sw) # inverse matrix of Sw

eigenData <- eigen(invSw %*% Sb)  # %*% product of matrices

J <- sum(diag(invSw %*% Sb))

# JUST TO RECALL :)
# J is the trace (sum of the diagonal elements) of the matrix
# invSw %*%Sb, which corresponds to the sum of the discriminating
# capacities for all discriminant axes, also called discriminant
# power or discriminatory information of X, which we want to maximize.

# The eigenvector associated with the largest eigenvalue of inv(Sw)Sb
# defines the linear combination of the columns of the (centred) matrix X,
# maximizing the ratio var(Xb a)/ var (Xw a) = (a^T Sb a) / (a^T Sw a),
# (between class variability / within class variability),
# i.e., defines the first discriminant axis
# and the corresponding eigenvalue is the value of the ratio,
# called discriminating capacity of the axis.

# The eigenvector associated with the 2nd largest eigenvalue of invSw %*% Sb
# defines the second discriminatant axis, i.e.,
# with the second largest discriminating capacity that is uncorrelated
# with the first discr. axis (but in general not orthogonal to LDA1, only
↪W-orthogonal).
# The corresponding eigenvalue is the 2nd largest value of the ratio
# intergroup variance / intragroup variance.

# And so on...

# The number of discriminant axes is, at most, the number of classes minus one.

# Loadings
loadings.LDA <- Re(eigenData$vectors[,1:(k-1)]) # similar to PCA

dim(loadings.LDA) <- c(p,k-1)

colnames(loadings.LDA) <- paste0(rep("LD",k-1),1:(k-1))
# to name the discr axes as LD1, LD2, ...

rownames(loadings.LDA) <- colnames(df.train)
#  to name the rows of loading matrix with the names of the variables

# discriminating capacities
discrCapacities <- Re(eigenData$values[1:(k-1)])
names(discrCapacities) <- paste0(rep("LD",k-1),1:(k-1))
prop.trace <- discrCapacities/sum(discrCapacities)

# discriminant scores
scores.LDA <- Xc %*% loadings.LDA # similar to PCA
```

```
    colnames(scores.LDA) <- paste0(rep("LD",k-1),1:(k-1))

    rownames(scores.LDA) <- rownames(df.train)
    # to name the rows of scores matrix with the names of the individuals

    bar.Z <- matrix(NA,k-1,k)
    # column j of this matrix will contains the centroid vector bar.zj
    # of the scores of the elements of class j

    for (j in 1:k){
        Xj = scores.LDA[cls==lev[j],]
        dim(Xj) <- c(sum(cls==lev[j]),k-1)
        bar.Z[,j] <- colMeans(Xj)
    }

    scores.test <- X.test.c %*% loadings.LDA
    dim(scores.test) <- c(M,k-1)
    colnames(scores.test) <- paste0(rep("LD",k-1),1:(k-1))
    rownames(scores.test) <- rownames(df.test)

    pred.test <- rep(NA,M)

    for (i in 1:M){
        scores.test.aux <- matrix(rep(scores.test[i,],k),k-1,k)
        dif.aux <- scores.test.aux - bar.Z
        dif.cls <- colSums(dif.aux^2)
        ord <- order(dif.cls)
        pred.test[i] <- lev[ord[1]]
    }

   # The function returns a list consisting of several components:

    out<-list(discr.load = loadings.LDA, # loadings defining the discr. axes

            discr.cap = discrCapacities, # resp. discr. capacities

            trace = J,  # total discriminatory information = sum discr.cap

            prop.trace = prop.trace, # discr.cap / J - analog to explained variance

            scores.train = scores.LDA,
            # coord of the N training individ in the discr. axes

            scores.test = scores.test,
            # coord of the M testing individ in the discr. axes

            pred.class = pred.test # predicted classes for the M testing individuals
           )
    return(out)
}
```

7.1 Applying the classifier `L.discr` to predict the class of a new vector

We are going to predict again the class of the hypothetical iris flower with petal and sepal measurements $\mathbf{y} = (6.2, 3.2, 5, 1.9)$, but this time applying the function `L.discr()`, which uses the scores of $\mathbf{y}$ on both discriminant axes LDA1 and LDA2 to predict the class of $\mathbf{y}$.

[124]:
```
df.train<-iris[,-5]
cls.train<-as.integer(iris[,5])
df.test<-c(6.2,3.2,5,1.9)

res.y<-L.discr(df.train,cls.train,df.test)
res.y
```

$discr.load A matrix: 4 × 2 of type dbl

	LD1	LD2
Sepal.Length	-0.2087418	-0.006531964
Sepal.Width	-0.3862037	-0.586610553
Petal.Length	0.5540117	0.252561540
Petal.Width	0.7073504	-0.769453092

$discr.cap **LD1** 32.1919291982779 **LD2** 0.285391042623112

$trace 32.477320240901

$prop.trace **LD1** 0.991212604965366 **LD2** 0.00878739503463399

$scores.train A matrix: 150 × 2 of type dbl

	LD1	LD2
1	-2.029033	-0.081417500
2	-1.794183	0.213194170
3	-1.885077	0.071922298
4	-1.714780	0.181748858
5	-2.046779	-0.139425359
6	-1.938464	-0.396143467
⋮	⋮	⋮
145	1.7233769	-0.65827402
146	1.4207621	-0.45468101
147	1.3036182	0.09850598
148	1.2503053	-0.22253869
149	1.4814539	-0.63554694
150	1.1786792	-0.08998504

$scores.test A matrix: 1 × 2 of type dbl

LD1	LD2
1.05415	-0.3114682

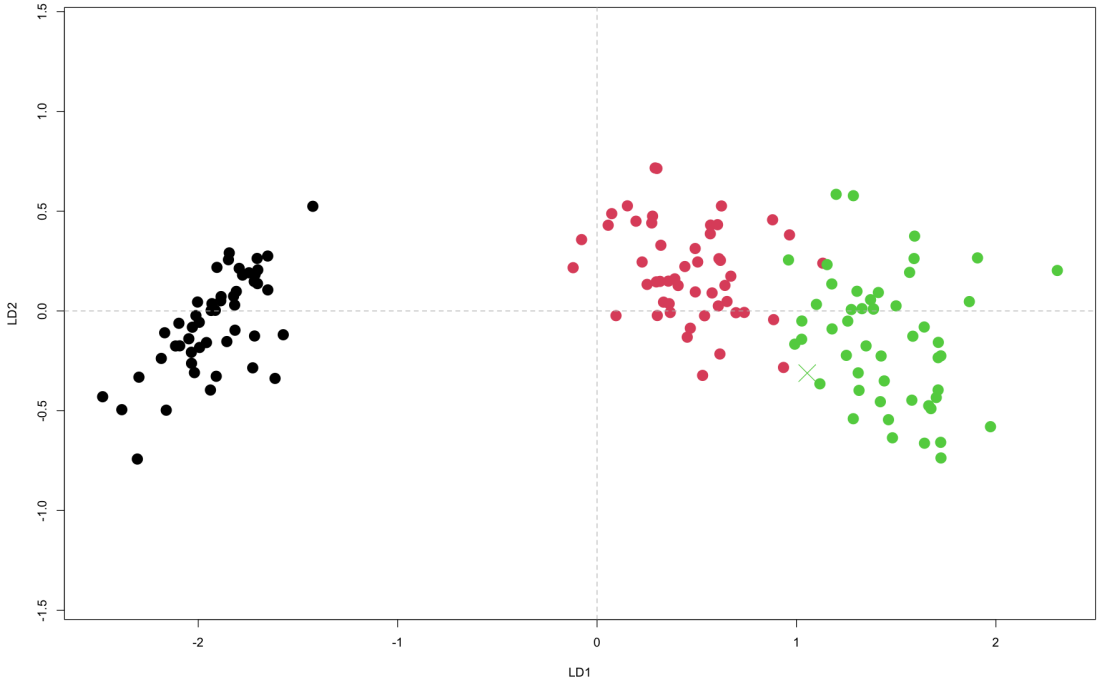
$pred.class 3

Note that the function `L.discr()` always centers the train data and computes the (new) coordinates of the testing data w.r.t. the mean vector (center of gravity) of the training data $\mathbf{X}^G$ before computing the discriminant axes, etc...

We can project the training individuals and the new guy $\mathbf{y} = (6.2, 3.2, 5, 1.9)$ on the plane defined LDA1 and LDA2 using the color corresponding to the respective classes with the code below. Intuively, the new element $\mathbf{y}$ (marked with an X) was correctly assigned to class 3 (virginica species).

[128]:
```
par(mfrow=c(1,1))
options(repr.plot.width =15, repr.plot.height = 10) ## windows dimensions
plot(res.y$scores.train,pch=16,cex=2,asp=TRUE,col=cls.train)
abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")
```

```
points(res.y$scores.test[,1],res.y$scores.test[,2],cex=3,pch=4,col=res.y$pred.class)
```



7.2 Behavior of LDA under linear changes of scale on each variable

Linear changes of scale on each variable, e.g., reduce the variable to unit variance or change the units of measurement, preserve the LDA discriminant power, i.e., the axes discriminating capacities (and henceforth also the trace proportions), and multiply the discriminant scores on the discriminants axes by scalars, thus preserving the shape of the cloud of the discriminant scores on the subspace defined by the discrimiant axes (up to axes scalling and/or reorientation), which strongly constrasts with PCA behavior.

As an example we are going to evaluate the effect of changing the units of measurements in the LDA outcomes, applying our LDA classifier to a modified version of iris dataset, where sepal and petal lengths are, respectively, measured in `mm` and `m` (instead of `cm`).

[132]:
```
## Original iris datasets and the testing y vector

head(iris) ## original iris dataset with all variables in cm  - just to remember
y=c(6.2,3.2,5,1.9)
y
```

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

1. 6.2 2. 3.2 3. 5 4. 1.9

[134]:
```
## Modified versions of iris dataset and (testing) y vector
## to became expressed in the new units of measurements

iris.mod <- iris
iris.mod[,1] <- iris[,1]*10 ## sepal lengths in mm
iris.mod[,3] <- iris[,3]/100 ## petal lengths in m

head(iris.mod)
var(iris.mod[,-5])  ## covariance matrix

y.mod <- y*c(10,1,.01,1)
y.mod
```

A data.frame: 6 × 5

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
1	51	3.5	0.014	0.2	setosa
2	49	3.0	0.014	0.2	setosa
3	47	3.2	0.013	0.2	setosa
4	46	3.1	0.015	0.2	setosa
5	50	3.6	0.014	0.2	setosa
6	54	3.9	0.017	0.4	setosa

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	68.5693512	-0.424340045	0.1274315436	5.16270694
Sepal.Width	-0.4243400	0.189979418	-0.0032965638	-0.12163937
Petal.Length	0.1274315	-0.003296564	0.0003116278	0.01295609
Petal.Width	5.1627069	-0.121639374	0.0129560940	0.58100626

1. 62 2. 3.2 3. 0.05 4. 1.9

[136]:
```
## Applying the L.discr function to the modified data

df.train <- iris.mod[,-5]
cls.train <- as.integer(iris.mod[,5])
df.test <- y.mod

res.mod <- L.discr(df.train,cls.train,df.test) ## testing data = training data
```

[138]:
```
## Comparing the discriminating capacities and the discriminant scores outcomes
## with the original vs modified data

# The discriminating capacities are the same
res.y$discr.cap ; res.mod$discr.cap

## The vectors of discriminant scores on each discriminant axis are proportional
head(res.y$scores.train / res.mod$scores.train)
```

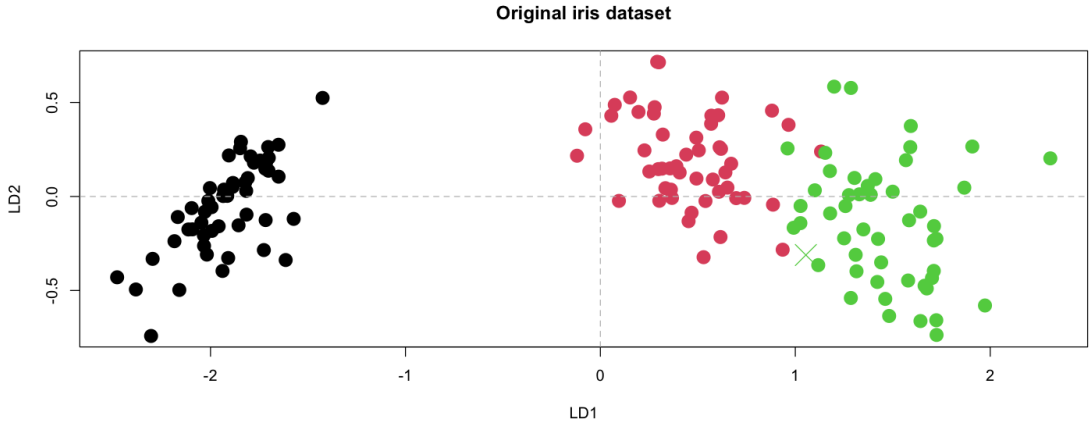
LD1 32.1919291982779 **LD2** 0.285391042623112

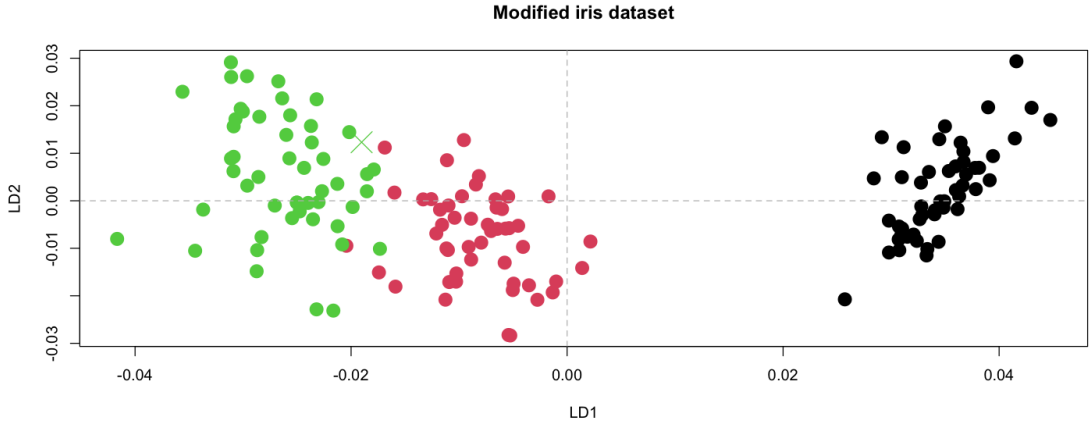
LD1 32.1919291982779 **LD2** 0.28539104262312

A matrix: 6 × 2 of type dbl

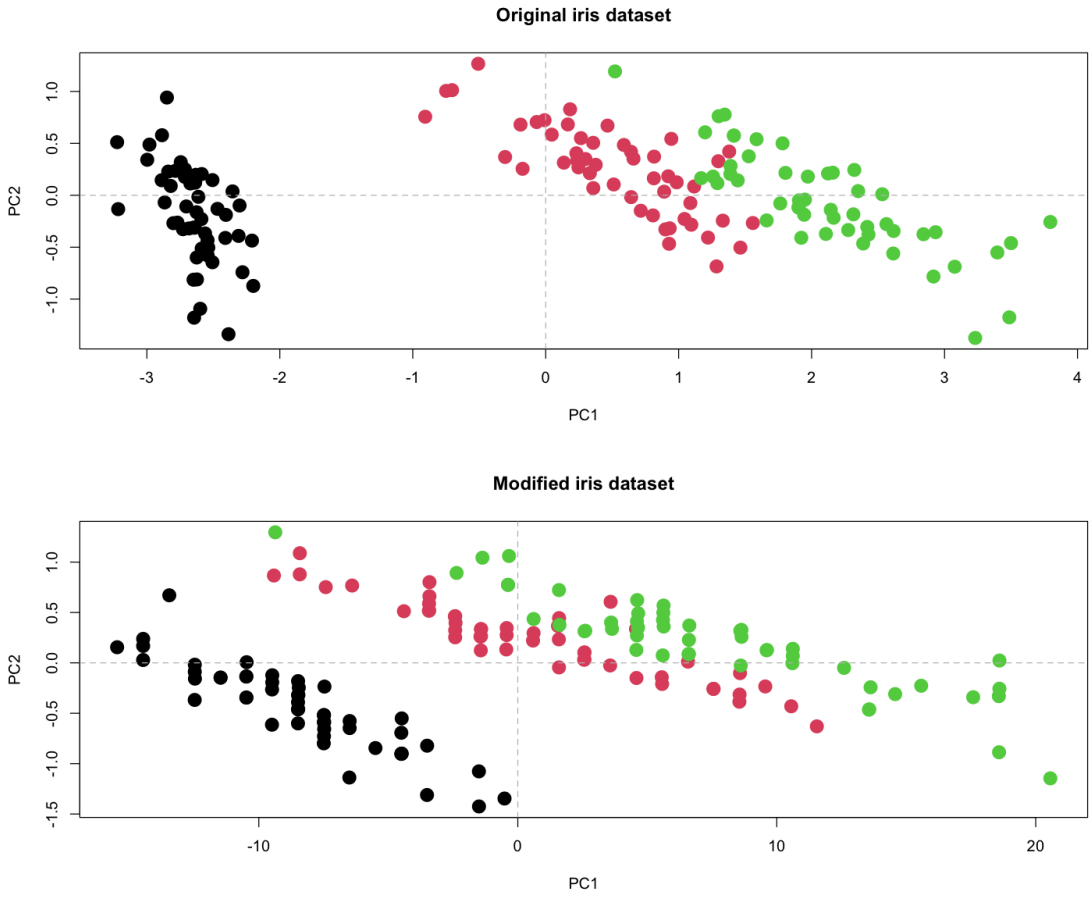
	LD1	LD2
1	-55.40704	-25.27468
2	-55.40704	-25.27468
3	-55.40704	-25.27468
4	-55.40704	-25.27468
5	-55.40704	-25.27468
6	-55.40704	-25.27468

[140]:
```
## Comparision of the shapes of the clouds of scores of
## iris flowers on the plane defined by LDA1 and LDA2
## obtained with the original vs modified data


par(mfrow=c(2,1))
options(repr.plot.width =12, repr.plot.height = 10) ## to define the plot windows␣
↪dimensions

plot(res.y$scores.train,pch=16,cex=2,col=cls.train, main="Original iris dataset")
abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")
points(res.y$scores.test[,1],res.y$scores.test[,2],cex=3,pch=4,col=res.y$pred.class)

plot(res.mod$scores.train,pch=16,cex=2,col=cls.train, main="Modified iris dataset")
abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")
points(res.mod$scores.test[,1],res.mod$scores.test[,2],cex=3,pch=4,col=res.mod$pred.
↪class)

## The clouds shapes are the same (up to the axes rescalling and reorientation)
```





[142]:
```
## To show that above type of LDA behavior contrasts with the PCA behavior
## we are going to compare the shapes of the clouds of scores of iris flowers
## on the plane defined by PC1 and PC2 obtained with the original vs modified
## data (no testing vector y is involved in this case).

iris.acp<-prcomp(iris[,-5])
iris.acp.mod<-prcomp(iris.mod[,-5])

par(mfrow=c(2,1))
options(repr.plot.width =12, repr.plot.height = 10) ## plot windows dimensions

plot(iris.acp$x,pch=16,cex=2,col=cls.train, main="Original iris dataset")
abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")

plot(iris.acp.mod$x,pch=16,cex=2,col=cls.train, main="Modified iris dataset")
abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")

## The shapes of the clouds are very different
## (even up to the axes rescalling and reorientation)
```



7.3 Evaluating the classifier L.discr on a testing set

To further explore our classifier `L.discr()` we are going to use as training data a random sample of 75 flowers, containing 25 of each one of the iris species setosa, versicolor and virginica. The remaining 75 iris flowers will be used as testing data.

[146]:
```
X1=sort(sample(1:50,25))
X2=sort(sample(51:100,25))
X3=sort(sample(101:150,25))


df.train<-iris[c(X1,X2,X3),1:4]  ## training individuals

cls.train<-as.integer(iris[c(X1,X2,X3),5])  ## resp. training classes

df.test<-iris[-c(X1,X2,X3),1:4]  ## testing individuals

cls.test<-as.integer(iris[-c(X1,X2,X3),5]) ##  actual testing classes


res<-L.discr(df.train,cls.train,df.test)
```

```
res$discr.cap

head(res$discr.load)

head(res$prop.trace)

head(res$scores.test)
```

LD1 31.3605590857972 **LD2** 0.389604717828349

A matrix: 4 × 2 of type dbl

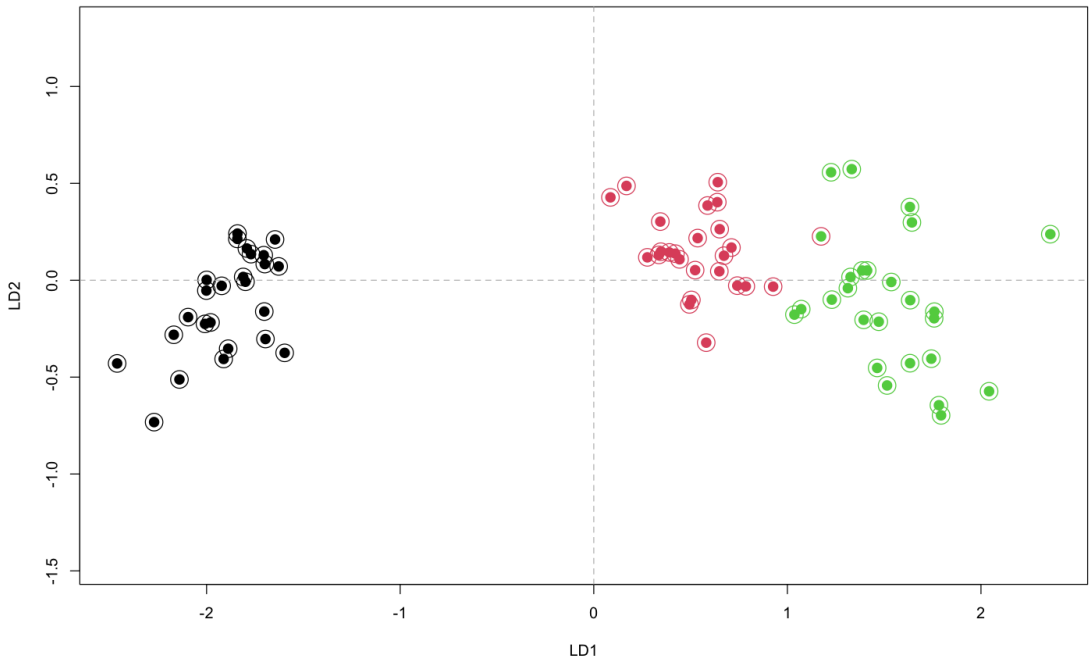
	LD1	LD2
Sepal.Length	-0.2195422	0.01843538
Sepal.Width	-0.3583460	-0.55702497
Petal.Length	0.5710014	0.26406974
Petal.Width	0.7052282	-0.78717882

LD1 0.987729048573165 **LD2** 0.0122709514268351

A matrix: 6 × 2 of type dbl

	LD1	LD2
2	-1.791878	0.1638469
4	-1.704750	0.1290208
6	-1.911815	-0.4064727
9	-1.646273	0.2103317
10	-1.841136	0.2132693
11	-2.095391	-0.1904459

[148]:
```
### To display the testing data with the actual and predicted classes

par(mfrow=c(1,1))
options(repr.plot.width =12, repr.plot.height = 8) ## to plot windows dimensions

plot(res$scores.test,pch=1,asp=TRUE,cex=2.5,col=cls.train)
### emptied dots - actual testing classes

abline(v=0,lty=2,col="gray")
abline(h=0,lty=2,col="gray")

points(res$scores.test[,1],res$scores.test[,2],col=res$pred.class,cex=1.5,pch=16)
### filled dots - predicted  testing classes
```



To evaluate the classification produced by `L.discr` we can compare the actual classes with the predict classes using a table similar to a confusion matrix. Several metrics including *Accuracy*, *Precision*, *Recall*, etc... can also be used to give further insight...

[151]:
```
#confusion matrix
actual <- factor(cls.test)
predicted <- factor(res$pred.class)

table(actual,predicted)
```

```
      predicted
actual  1  2  3
     1 25  0  0
     2  0 24  1
     3  0  0 25
```

8 `lda` function (MASS)

We are going now to give a brief account on the `lda` function from the MASS library and compare some of its outcomes with the results produced by our classifier `L.discr`.

[154]:
```
## Applying the lda function to iris dataset

library(MASS)

lda.train  <- iris[,-5]
```

```
lda.groups <- as.factor(as.integer(iris[,5]))

lda.res <- lda(lda.groups ~ . ,data=lda.train)
## the iris groups is the response variable!

lda.res
```

```
Call:
lda(lda.groups ~ ., data = lda.train)

Prior probabilities of groups:
        1         2         3
0.3333333 0.3333333 0.3333333

Group means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1        5.006       3.428        1.462       0.246
2        5.936       2.770        4.260       1.326
3        6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776 -0.02410215
Sepal.Width   1.5344731 -2.16452123
Petal.Length -2.2012117  0.93192121
Petal.Width  -2.8104603 -2.83918785

Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

The discriminating capacities produced by `lda` are the squared singular values and can be retrieved as follows:

[157]:
```
## Discriminanting capacities - lda (MASS)
lda.eig <- lda.res$svd^2
lda.eig
```

1. 2366.10679607343 2. 20.9762416327959

To obtain the discriminant scores and the predicted classes we choose, respectively, the components "x" and "class" from the list produced with the predict() function called with the object returned by the `lda` function.

[160]:
```
# Matrix of scores - lda (MASS)
pred.iris <- predict(lda.res)
scores.iris <- pred.iris$x
class.iris <- pred.iris$class
head(scores.iris)
head(class.iris)
```

A matrix: 6 × 2 of type dbl

	LD1	LD2
1	8.061800	-0.3004206
2	7.128688	0.7866604
3	7.489828	0.2653845
4	6.813201	0.6706311
5	8.132309	-0.5144625
6	7.701947	-1.4617210

1. 1 2. 1 3. 1 4. 1 5. 1 6. 1

Levels: 1. '1' 2. '2' 3. '3'

The matrix of loadings can be obtained as follows:

[163]:
```
# Matrix of loadings - lda (MASS)
lda.res$scaling
```

A matrix: 4 × 2 of type dbl

	LD1	LD2
Sepal.Length	0.8293776	-0.02410215
Sepal.Width	1.5344731	-2.16452123
Petal.Length	-2.2012117	0.93192121
Petal.Width	-2.8104603	-2.83918785

Remark on the normalization of the loadings matrix According to the `lda` help function the eigenvectors of $\mathbf{S}_w^{-1}\mathbf{S_b}$ defining the matrix of loadings $\mathbf{V}$ are chosen in such a way that the matrix "transforms observations to discriminant functions, normalized so that within groups covariance matrix is spherical".

We can compute the within-class covariance of the scores of the iris flowers produced by the `lda` function as we did to define the matrix of within-class variability $\mathbf{S}_w$:

[167]:
```
Zw = scores.iris
for (j in 1:k){
    Zj <- Zw[lda.groups==lev[j],]

    Zjc <- scale(Zj,scale=FALSE)

    Zw[lda.groups==lev[j],] <- Zjc
}

round(cov(Zw),7)
```

A matrix: 2 × 2 of type dbl

	LD1	LD2
LD1	0.9865772	0.0000000
LD2	0.0000000	0.9865772

The within groups covariance matrix is spherical since the covariance matrix is scalar. This normalization is equivalent to compute each distance between the discriminant score of a vector and the centroid of the scores of a class using the Mahalanobis distance defined by the class covariance matrix, thus accounting for the scores distribution within each class.

`lda` vs `L.discr` Before comparing some results produced by `lda` (MASS) function and our classifier `L.discr`, we make some remarks about eigenvalues and eigenvectors:

Let $\mathbf{v}$ be an eigenvector of a matrix $\mathbf{A}$ associated with an eigenvalue λ, that is,

$$\mathbf{A}\,\mathbf{v} = \lambda\mathbf{v}.$$

For any $\alpha \neq 0$ we have the following:

- $\mathbf{u} = \alpha\mathbf{v}$ is also an eigenvector of $\mathbf{A}$ associated with the same eigenvalue λ. Actually,

$$\mathbf{A}\,\mathbf{u} = \mathbf{A}\,\alpha\mathbf{v} = \alpha\mathbf{A}\,\mathbf{v} = \alpha\,\lambda\mathbf{v} = \lambda\mathbf{u}.$$

  Hence eigenvectors multiplied by nonzero scalars are still eigenvectors associated with the same eigenvalue.

- $\mathbf{v}$ is an eigenvector of $\mathbf{B} = \alpha\mathbf{A}$ associated with the eigenvalue $\mu = \alpha\,\lambda$. Actually,

$$\mathbf{B}\,\mathbf{v} = \alpha\mathbf{A}\,\mathbf{v} = (\alpha\,\lambda)\mathbf{v} = \mu\mathbf{v}.$$

  Hence nonzero proportional matrices have the same eigenvectors and the corresponding eigenvalues verify the same proportional relation.

The matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$ associated with `lda`(MASS) multiplied by $\frac{k-1}{n-k}$ equals the matrix $\mathbf{S}_w^{-1}\mathbf{S}_b$ associated with L.discr function (see the slides of Prof. Cadima). Hence:

- The discriminating capacities produced by `lda` multiplied by $\frac{k-1}{n-k}$ are equal to the discriminating capacities obtained with L.discr. The proportions of the trace are obviously the same.
- Both functions have the same vectors of loadings, since $\mathbf{S}_w^{-1}\mathbf{S}_b$ has the same eigenvectors for both functions. Nevertheless distinct normalizations can be used to define the scores of the individuals projected on the discriminant axes, possibly leading to distinct predicted classes. For instance, L.discr does not use Mahalanobis distances, i.e., does not account for the distribution of the scores within each class...
- The vectors of scores of the individuals projected on each discriminant axis by both functions are proportional and therefore the clouds of scores produced by both functions have the same shape (up to axes scalling and / or reorientation).

res<-L.discr(iris[,-5],as.numeric(iris[,5])) res$pred.class

Let us verify, for instance, that the discriminanting capacities produced by lda (MASS) multiplied by $\frac{k-1}{n-k}$ are equal to the discriminating capacities produced by the `L.discr`:

[176]:
```
N<-nrow(lda.train)
k<-length(unique(lda.groups))
N ; k
res<-L.discr(lda.train,lda.groups)
lda.res$svd^2 *(k-1)/(N-k) ; res$discr.cap  ### should be equal
```

150

3

1. 32.191929198278 2. 0.285391042623073

LD1 32.1919291982779 **LD2** 0.285391042623112

We finish this section comparing the iris classes predicted by both functions and also with the groud truth:

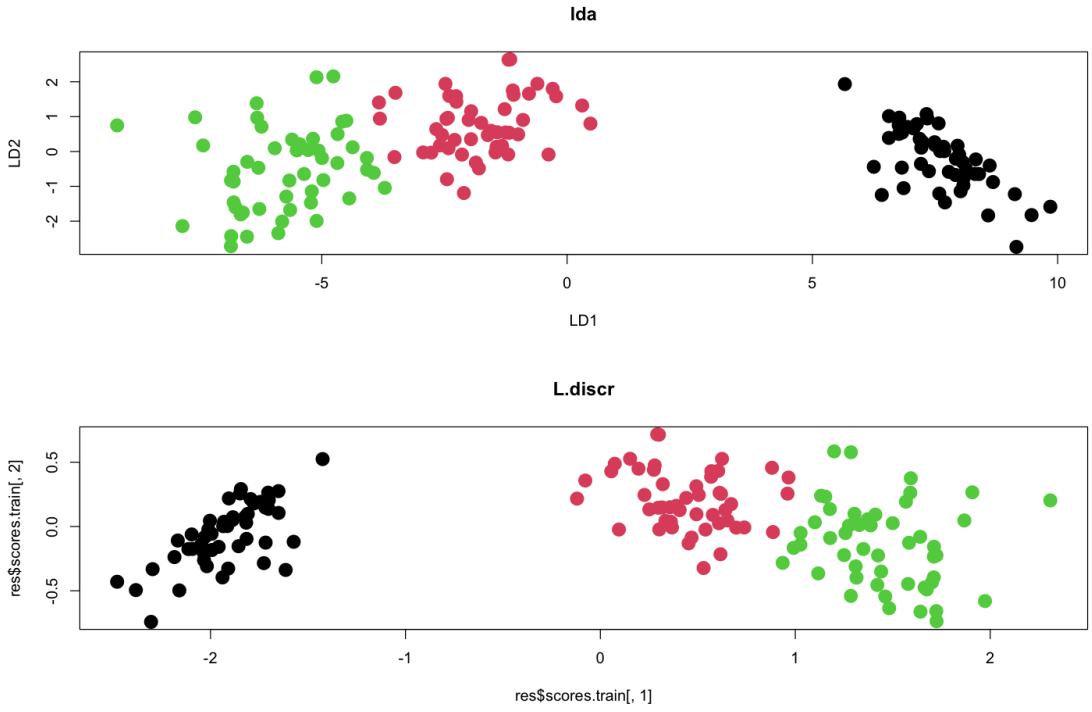
[179]:
```
# Confusion matrices

actual <- lda.groups
pred.lda <- pred.iris$class
pred.our <- res$pred.class

table(pred.lda,pred.our) ### Both functions predict the same species for all flowers
table(actual,pred.lda)  ### Both functions mislabeled 3 iris flowers
```

```
        pred.our
pred.lda  1  2  3
       1 50  0  0
       2  0 49  0
       3  0  0 51
```

```
      pred.lda
actual  1  2  3
```

```
  1 50  0  0
  2  0 48  2
  3  0  1 49
```

[181]:
```
## Comparison of the shapes produced by both functions,
## of the projected clouds in the discriminant plane
## defined by LDA1 and LDA2

par(mfrow=c(2,1))
plot(scores.iris,pch=16,col=class.iris,cex=2, main="lda")
plot(res$scores.train[,1],res$scores.train[,2],pch=16,col=res$pred.class,cex=2,main="L.
↪discr")
```



9 APPENDIX

Let us prove that the maximum of the FDR

$$\frac{\mathbf{a}^T \mathbf{S}_b \mathbf{a}}{\mathbf{a}^T \mathbf{S}_w \mathbf{a}} \quad \text{with} \quad \mathbf{a} \neq \vec{0}, \qquad (1)$$

is attained along the direction of an eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$ associated with its largest eigenvalue that we denoted λ_1.

The ratio above is invariant under multiplication of $\mathbf{a}$ by nonzero scalars. Thus we can normalize the denominator in order that $\mathbf{a}^T\mathbf{S}_w\mathbf{a} = 1$, reducing the problem of maximizing (1) to the problem of maximizing the between-class variance,

$$\mathbf{a}^T \, \mathbf{S}_b \, \mathbf{a}, \qquad (2)$$

subject to the addtional constraint that $\mathbf{a}$ is a $\mathbf{S}_w$-unit vector, that is,

$$\mathbf{a}^T\,\mathbf{S}_w\,\mathbf{a} \;=\; 1. \qquad (3)$$

In order to maximize the between-class vaariance (2) subject to (3) we consider the vector

$$\mathbf{b} = \mathbf{S}_w^{\frac{1}{2}}\,\mathbf{a}$$

Since $\mathbf{S}_w$ is symmetric, it is also diagonalizable and we can find an invertible matrix $\mathbf{P}_w$ and a diagonal matrix $\mathbf{D}_w$ such that

$$\mathbf{S}_w = \mathbf{P}_w\,\mathbf{D}_w\mathbf{P}_w^{-1}.$$

Assuming $\mathbf{S}_w$ invertible (which implies $k \leq N-p$), all diagonal entries of $\mathbf{D}_w$ are positive real numbers and we can consider the square root of $\mathbf{S}_w$,

$$\mathbf{S}_w^{\frac{1}{2}} = \mathbf{P}_w\,\mathbf{D}_w^{\frac{1}{2}}\mathbf{P}_w^{-1},$$

where $\mathbf{D}_w^{\frac{1}{2}}$ denotes the diagonal matrix containing the square roots of the diagonal elements of $\mathbf{D}_w$ (verify that $(\mathbf{S}_w^{\frac{1}{2}})^2 = \mathbf{S}_w$).

Now consider the auxiliary vector $\mathbf{b} = \mathbf{S}_w^{\frac{1}{2}}\,\mathbf{a}$. Then $\mathbf{a} = \mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{b}$ and $\mathbf{a}^T = \mathbf{b}^T\,\mathbf{S}_w^{-\frac{1}{2}}$. Since

$$1 \;=\; \mathbf{a}^T\,\mathbf{S}_w\,\mathbf{a} \;=\; \mathbf{b}^T\,\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_w\,\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{b} \;=\; \mathbf{b}^T\,\mathbf{b},$$

maximize (2) subject to (3) is equivalent to maximize

$$\mathbf{b}^T\,\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_b\,\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{b} \qquad (3)$$

subject to

$$\mathbf{b}^T\,\mathbf{b} \;=\; 1. \qquad (4)$$

Since $\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_b\,\mathbf{S}_w^{-\frac{1}{2}}$ is symmetric its eigenvalues/eigenvectors are real numbers/vectors and we know that the maximum of (3) subject to (4) occurs when $\mathbf{b}$ is a (unit) eigenvector of $\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_b\,\mathbf{S}_w^{-\frac{1}{2}}$ associated with its largest eigenvalue denoted λ_1 (see the section devoted to PCA), that is,

$$\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_b\,\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{b} = \lambda_1\,\mathbf{b}.$$

Since $\mathbf{b} = \mathbf{S}_w^{\frac{1}{2}}\mathbf{a}$, the previous relation is equivalent to

$$\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_b\,\mathbf{S}_w^{-\frac{1}{2}}\,\mathbf{S}_w^{\frac{1}{2}}\,\mathbf{a} = \lambda_1\,\mathbf{S}_w^{\frac{1}{2}}\mathbf{a}$$

that is, multiplying on the left both members of the equation by $\mathbf{S}_w^{\frac{1}{2}}$,

$$\mathbf{S}_w^{-1}\mathbf{S}_b\,\mathbf{a} = \lambda_1\,\mathbf{a}.$$

Hence $\mathbf{a}$ is an eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$ associated with its largest eigenvalue λ_1, as we intend to prove.

[ ]:

10 CHALLENGE

Modify the code of the function `L.discr` to accept an additional (logical) argument that indicates if the Mahalanobis distance should be employed to compute the distance between each score of an element of the testing data and the centroid of the scores of each class of training data, and adapt the function's code accordingly. You can use the function below to compute the Mahalanobis distances.

The following function accepts 2 vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, and an invertible matrix $\mathbf{S}$ of order d, and computes the Mahalanobis distance between $\mathbf{x}$ and $\mathbf{y}$ w.r.t. the invertible covariance matrix $\mathbf{S}$, i.e., it computes

$$d_{\mathbf{S}}(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{y}).$$

[193]:
```r
### computes the Mahalanobis distance between x,y,
### assocociated with an invertible covariance matrix S
d.Maha <- function(x,y,S){
t(x-y) %*% solve(S) %*% (x-y)
}
```

[ ]:

[ ]: