# Contents

# List of Definitions



# List of Results

# 1 Introduction

Multivariate statistics focuses on the situation where several random variables (usually related) are observed for a single individual. For instance, recording the examination marks for all students taking this course forms a series of *univariate* observations, whereas recording the examination marks that each student obtained in five different courses would constitute *multivariate* observations. As you can see, multivariate observations are the norm rather than the exception, we rarely measure one single variable in isolation!

Suppose that we have $p$ variables and $n$ individuals. Many classical multivariate statistical methods focus on the case where $p < n$, but, through modern research, many of these classical techniques have been extended to the case where $p$ is (possibly much) greater than $n$.

The general topic of multivariate statistics contains a very broad range of techniques. Iin this course we will only be able to cover a selected subset of these techniques. However, the intuitive ideas and chosen examples that we will see form the basis from which many other methods arise, so at the end of the course you should have a good notion of what multivariate statistics can do for you.

Below is one possible classification of sub-topics within multivariate statistics, with some examples.

1. **Exploratory data analysis**

   The aim is to explore large datasets to get an informal idea of the kind of information that they contain. One is usually interested in producing plots or tables that summarize multivariate information in an easily interpretable manner, or that highlight certain features that are not easy to detect by staring at the data or using univariate/bivariate methods (histograms, contingency tables, scatter plots etc.).

   For instance, suppose a group of students are asked to rate how important several characteristics are in order to have a good lecture (e.g. good notes, well-organized, knowledgeable professor etc.). Suppose we ask a class of $n = 50$ students about $p = 10$ such characteristics. Are some of these characteristics consistently ranked high/low by most students? Are there groups of students giving overall similar answers? If we also asked professors these same questions, can we get an overall view of how they compare with students?

   As usual, exploratory analysis often serves as a first look at the data, to help decide how to perform subsequent analyses (for instance, those listed in the following sections).

2. **Inferential statistics**

   The goal is to learn (or infer) the characteristics of the population from which the observed sample was obtained, while characterizing uncertainty. For instance, one might perform a multivariate test to compare several means across

$K$ groups (as opposed to univariate tests such as the F-test that compare a single outcome across $K$ groups). Another example is to jointly regress a set of response variables on a set of predictors, which is an extension of the univariate linear model where we study the effect of a set of predictors on a single response.

One more example is to learn about the correlation structure between variables, and then compare this structure across two groups. As an illustration, suppose that in normal conditions genes A, B and C interact with each other, so that their expressions are correlated. A scientist might be interested in studying what happens in cancer patients: changes in the correlation structure could indicate that the disease interrupted the usual interactions between these genes.

3. **Classification**

   The objective is to classify an individual into one of $K$ groups, based on a sample of individuals (called the *training sample*) for which we do know the group labels. As opposed to clustering problems, here groups are known in advance.

   For instance, the online movie streaming company Netflix announced the so-called Netflix contest a few years ago. The goal was to determine which movies should be recommended to a given individual. To simplify, suppose Netflix ratings for a given film range from 1 star (hated the film) to 5 stars (loved it), hence here we would have $K = 5$ classes. Given the ratings that an individual gave to previously viewed films, can we predict his rating for a future film?

4. **Clustering (or Class Discovery)**

   The goal is to find groups of individuals (or groups or variables) that behave in a similar manner. For instance, suppose we collect $p = 10$ macro-economical variables for $n = 50$ countries. Clustering could help detect two subsets of countries that behave in a similar manner, e.g. developed vs. developing economies.

   The key in cluster analysis is that one typically does not know which are these groups, or if they even exist, and hopes that a clustering analysis will help reveal them. For instance, doctors suspect that not all patients diagnosed with Alzheimer's Disease suffer the same variety of the disease, but they do not have any clinical test to tell these suspected varieties appart. An approach that has been taken is to study the complete genome of a sample of individuals, and then group patients with similar genome characteristics (in some sense) in the hopes that these groups would correspond to disease subtypes.

# 2 Algebra review

Humans have difficulties in processing data that have more than a few dimensions (for most of us, 3 dimensions in space or perhaps 3 dimensions in space + time is as good as it gets). As the number of dimensions (measured variables) grows, our intuition may fail and it is helpful to use some of the tools provided by linear algebra. Here we review some notions that are basic but can be tremendously useful, and will serve as a basis for much of the material we will cover during the course. It is important that you familiarize yourselves with these ideas, they will pay off in the very short run.

We start with some basic notation.

- $p, q \in \{1, 2, \ldots\}$ will always denote positive integers, $\mathbb{R}^p$ is the $p$-dimensional real space, and $\mathbb{R}^{p \times q}$ is the space of $p \times q$ real matrices. All matrices and vectors are *real* (unless explicitly specified).

- Vectors are in small boldface letter, e.g. $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, and are *always* column vectors, unless explicitly specified. For instance,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}.$$

  The $i$-th entry of $\mathbf{x}$ is typically written $x_i$.

- We denote matrices with uppercase letters, e.g. $V, W, A, B$ and their corresponding $(i, j)$ element with lowercase letters, so for instance if $V$ is a $p \times p$ matrix, $v_{ij}$ is its $(i, j)$ element, sometimes also written $(V)_{ij}$.

- We indicate the transpose of $V$ with $V^\mathsf{T}$. If $V = V^\mathsf{T}$, we say that $V$ is symmetric, or self-adjoint. The transpose of a (column) vector $\mathbf{x}$ is written $\mathbf{x}^\mathsf{T}$. It is a *row* vector with the same elements as $\mathbf{x}$.

- $\mathbf{1} = (1, 1, \ldots, 1)^\mathsf{T}$ will denote the (column) vector and its dimension will be implicitly determined by the context; if we wish to emphasize its dimension, we will write $\mathbf{1}_p \in \mathbb{R}^p$. We shall denote by $I$ the (square) identity matrix, with 1s on the diagonal and zero everywhere else. Its dimension is implicitly defined, unless we write $I_q$, which means it is the $q \times q$ identity matrix.

- Constants will be denoted by Greek letters, e.g. $\alpha, \beta, \lambda \in \mathbb{R}$.

- $\alpha\mathbf{x}$ is the same as $\mathbf{x} \in \mathbb{R}^p$, but with each entry multiplied by $\alpha$. Similarly for $\alpha V$, where $V$ is a matrix.

- $\mathbf{x} + \mathbf{y}$ means we add vectors $\mathbf{x}$ and $\mathbf{y}$ element-wise (of course $\mathbf{x}$ and $\mathbf{y}$ must have the same number of elements), $VW$ is the matrix product (supposing that the matrices $V, W$ have compatible dimensions).

- If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ are two vectors, then $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^p x_i y_i$ is the inner-product (or dot product) between them. Notice that $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\mathsf{T}\mathbf{y} \in \mathbb{R}$, but $\mathbf{x}\mathbf{y}^\mathsf{T} \in \mathbb{R}^{p \times p}$.
- $|\mathbf{x}|$ will denote the Euclidean norm of $\mathbf{x} \in \mathbb{R}^p$, i.e. $|\mathbf{x}| = \sqrt{\sum_{i=1}^p x_i^2}$.

## 2.1 Matrices and Eigendecompositions

Let $A$ and $B$ be $p \times p$ matrices, denote their determinants by $\det(A)$ and $\det(B)$ and their inverses by $A^{-1}, B^{-1}$. Also define the *trace* of $A$ by $\mathrm{Tr}(A) = \sum_{i=1}^p a_{ii}$.

**Theorem 2.1.1 (Basic results about Determinants and Traces).** We have the following identities:

1. $(A^\mathsf{T})^{-1} = (A^{-1})^\mathsf{T}$,
2. $(AB)^{-1} = B^{-1}A^{-1}$,
3. $\det(A) = \det(A^\mathsf{T})$,
4. $\det(A^{-1}) = 1/\det(A)$,
5. $\det(AB) = \det(A)\det(B)$,
6. $\det(\alpha A) = \alpha^p \det(A)$,
7. $\mathrm{Tr}(A + B) = \mathrm{Tr}(A) + \mathrm{Tr}(B)$,
8. $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$. This also holds if $A, B$ are rectangular such that $AB$ is defined and is a square matrix.
9. $\mathrm{Tr}(B^{-1}AB) = \mathrm{Tr}(A)$.

*Proof.* Show $1, 2, 7, 8, 9$. □

We review the concept of positive definite matrices (sometimes called non-negative definite matrices), which we shall see later on is important for defining distances between points.

**Definition 2.1.2 (Positive definite matrices).** A square $p \times p$ matrix $A$ is said to be **positive definite** if and only if $\mathbf{x}^\mathsf{T}A\mathbf{x} > 0$ for any non-zero vector $\mathbf{x}$. When $\mathbf{x}^\mathsf{T}A\mathbf{x} \geq 0$ for all non-zero $\mathbf{x}$ the matrix is said to be **positive semi-definite** (or nonnegative definite). Similary, $A$ is **negative definite**, respectively **negative semi-definite**, if $\mathbf{x}^\mathsf{T}A\mathbf{x} < 0$, respectively $\mathbf{x}^\mathsf{T}A\mathbf{x} \leq 0$, for all non-zero $\mathbf{x}$.

We now review the concept of eigenvectors and eigenvalues, which will prove useful during the course, and their relation to positive definiteness.

**Definition 2.1.3 (Eigenvalues and eigenvectors).** Let $A$ be a $p \times p$ square matrix. A non-zero vector $\mathbf{v} = (v_1, \ldots, v_p)^\mathsf{T}$ is an eigenvector of A if and only if $A\mathbf{v} = \lambda\mathbf{v}$, and in this case $\lambda$ is called its eigenvalue.

In general, eigenvectors may either not exist or may involve complex numbers, but in this course we shall focus on certain matrices $A$ for which eigenvectors exist and have real entries.

Before proceeding we recall how to compute the *eigendecomposition* (eigenvalues and eigenvectors) of $A$, at least conceptually.

**Proposition 2.1.4 (Calculation of eigenvectors and eigenvalues).** $\lambda \in \mathbb{R}$ is an eigenvalue of $A$ if and only if $\det(A - \lambda I) = 0$, where $I$ is the identity matrix. In particular, if $\lambda$ is an eigenvalue of $A$, then the corresponding eigenvector $\mathbf{v}$ satisfies $(A - \lambda I)\mathbf{v} = \mathbf{0}$ (a vector of zeroes).

*Proof.* Assume $\mathbf{v}$ is a non-zero vector. Then

$$A\mathbf{v} = \lambda\mathbf{v} \quad \Leftrightarrow \quad (A - \lambda I)\mathbf{v} = \mathbf{0} \quad \Leftrightarrow \quad |A - \lambda I| = 0.$$

□

That is, to find eigenvalues we could in principle find the solutions of $\det(A - \lambda I) = \mathbf{0}$, which we denote $\lambda_1, \ldots, \lambda_p$ (not all of them need to be distinct). To find the corresponding $i^{th}$ eigenvector we solve the system of linear equations $(A - \lambda_i I)\mathbf{v}_i = \mathbf{0}$ (although the solution is not unique—see below).

In practice, eigendecomposition algorithms work differently; note in particular that given an eigenvector $\mathbf{v}_i$ it's trivial to find the corresponding eigenvalue $\lambda_i = \sqrt{(A\mathbf{v}_i)^\mathsf{T} A\mathbf{v}_i \,/\, (\mathbf{v}_i^\mathsf{T}\mathbf{v}_i)}$, but finding $\mathbf{v}_i$ given $\lambda_i$ is nontrivial.

In `R` we can find the eigendecomposition using the function `eigen`. Here are some fundamental results about eigenvalues and eigenvectors for symmetric matrices.

**Theorem 2.1.5 (Fundamental results about eigenvalues and eigenvectors).** Let $A \in \mathbb{R}^{p \times p}$ be symmetric.

1. $A$ has $p$ linearly independent eigenvectors with (not necessarily distinct) **real** eigenvalues $\lambda_1, \ldots, \lambda_p$. The *multiplicity* of an eigenvalue is the number of linearly independent eigenvectors that are associated to it.

2. The eigenvectors corresponding to different eigenvalues are orthogonal: $\mathbf{v}_i^\mathsf{T}\mathbf{v}_j = 0$ if $A\mathbf{v}_i = \lambda_i\mathbf{v}_i, A\mathbf{v}_j = \lambda_j\mathbf{v}_j$ and $\lambda_i \neq \lambda_j$.

3. $\det(A) = \prod_{i=1}^{p} \lambda_i$

4. $\mathrm{Tr}(A) = \sum_{i=1}^{p} \lambda_i$

5. If $\mathbf{v}_i$ is an eigenvector of $A$ with eigenvalue $\lambda_i$ and $\alpha \neq 0$, then any $\alpha\mathbf{v}_i$ is also an eigenvector and also has eigenvalue $\lambda_i$.

6. If $B = \alpha A$ where $\alpha$ is some constant, then $B$ has the same eigenvectors as $A$, and the $i^{\text{th}}$ eigenvalue of $B$ is $\alpha\lambda_i$.

7. If the inverse $A^{-1}$ exists, then it has the same eigenvectors, with corresponding eigenvalues $1/\lambda_1, \cdots, 1/\lambda_p$.

8. A symmetric matrix $A$ is positive definite if and only if all $\lambda_i > 0$, and negative definite if and only if all $\lambda_i < 0$. When all $\lambda_i \geq 0$, the matrix is positive semi-definite (or negative semi-definite if all $\lambda_i \leq 0$).

These properties are important because many multivariate methods are based on analysing covariance matrices, which are symmetric and positive definite (or semi-definite). The eigenvalues are then all real, and are traditionally ordered $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. Also, from property (5), if $\mathbf{v}_i \in \mathbb{R}^p$ then we can without loss of generality use the unit eigenvector

$$\mathbf{e}_i = \sqrt{\frac{1}{\mathbf{v}_i^\mathsf{T}\mathbf{v}_i}}\mathbf{v}_i$$

and define the matrix $E$ to be the matrix with columns $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p$. Note that $E^\mathsf{T}E = I$ $\big($and therefore $EE^\mathsf{T} = I$, since $(EE^\mathsf{T})E = E(E^\mathsf{T}E) = EI = E\big)$. Real square matrices with this property are said to be *orthogonal*.

**Remark 2.1.6 (Non-uniqueness of eigenvectors).** Notice that even when we restrict eigenvectors to have length 1, they are not uniquely defined (i.e. $-\mathbf{e}_i$ is also an eigenvector with length 1). Furthermore, if there are two linearly independent vectors that have the same eigenvalue, then any linear combination of them has the same eigenvalue. Eigenvectors are therefore not uniquely defined; but the subspace spanned by all eigenvectors corresponding to an eigenvalue is uniquely defined.

The eigenvectors and eigenvalues give us a useful way to re-construct the matrix. The following result is a major result of linear algebra. You should know it like the back of your hand.

**Theorem 2.1.7 (Spectral Decomposition Theorem).** Let $A$ be a $p \times p$ real symmetric matrix. Then there exists a real orthogonal matrix $E$ and a real diagonal matrix $\Lambda$ such that

$$A = E\Lambda E^T.$$

We see that the diagonal elements of $\Lambda$, denoted here $\lambda_1, \ldots, \lambda_p$, are eigenvalues of $A$ and the columns $\mathbf{e}_1, \ldots, \mathbf{e}_p$ of $E$ are orthonormal eigenvectors. An equivalent expression for $A$ is

$$A = \lambda_1\mathbf{e}_1\mathbf{e}_1^\mathsf{T} + \lambda_2\mathbf{e}_2\mathbf{e}_2^\mathsf{T} + \cdots + \lambda_p\mathbf{e}_p\mathbf{e}_p^\mathsf{T}.$$

Check that this result is indeed valid in light of Remark 2.1.6. Looking at the

expression, it becomes clear that eigenvectors with large $|\lambda_i|$ will have more weight in the reconstruction, hence in some sense they are more essential ingredients of $A$.

**Example 2.1.8** (Symmetric Matrix Approximation using Spectral Decomposition)**.** Let us take a matrix A, find its eigenvectors and eigenvalues and then successively reconstruct A. Below is the R code and output, where we store the reconstruction of A into a new matrix B. We see that with 3 eigenvectors B already gives a reasonably good approximation to A. This can also be seen visually in Figure 1.

```
A <- matrix(c(1,.9,.5,.1,.1, .9,1,.5,.1,.1, .5,.5,1,.5,.5, .1,.1,.5,1,.9,
              .1,.1,.5,.9,1), nrow=5, byrow=TRUE)
A

##      [,1] [,2] [,3] [,4] [,5]
## [1,]  1.0  0.9  0.5  0.1  0.1
## [2,]  0.9  1.0  0.5  0.1  0.1
## [3,]  0.5  0.5  1.0  0.5  0.5
## [4,]  0.1  0.1  0.5  1.0  0.9
## [5,]  0.1  0.1  0.5  0.9  1.0

l <- eigen(A)$values
v <- eigen(A)$vectors

op <- par(mfrow=c(3,2), mai=rep(.4,4), oma=rep(0,4))
image(A)
title(main="True matrix")
B <- l[1] * matrix(v[,1],ncol=1) %*% matrix(v[,1],nrow=1)
round(B,2)

##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.50 0.50 0.59 0.50 0.50
## [2,] 0.50 0.50 0.59 0.50 0.50
## [3,] 0.59 0.59 0.70 0.59 0.59
## [4,] 0.50 0.50 0.59 0.50 0.50
## [5,] 0.50 0.50 0.59 0.50 0.50

image(B, zlim=range(A))
title(main = 'Rank 1 approx')
B <- B + l[2] * matrix(v[,2],ncol=1) %*% matrix(v[,2],nrow=1)
round(B,2)

##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.92 0.92 0.59 0.07 0.07
## [2,] 0.92 0.92 0.59 0.07 0.07
## [3,] 0.59 0.59 0.70 0.59 0.59
## [4,] 0.07 0.07 0.59 0.92 0.92
## [5,] 0.07 0.07 0.59 0.92 0.92
```

```r
image(B, zlim=range(A))
title(main = 'Rank 2 approx')
B <- B + l[3] * matrix(v[,3],ncol=1) %*% matrix(v[,3],nrow=1)
round(B,2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.95 0.95  0.5 0.10 0.10
## [2,] 0.95 0.95  0.5 0.10 0.10
## [3,] 0.50 0.50  1.0 0.50 0.50
## [4,] 0.10 0.10  0.5 0.95 0.95
## [5,] 0.10 0.10  0.5 0.95 0.95
```

```r
image(B, zlim=range(A))
title(main = 'Rank 3 approx')
B <- B + l[4] * matrix(v[,4],ncol=1) %*% matrix(v[,4],nrow=1)
round(B,2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.95 0.95  0.5  0.1  0.1
## [2,] 0.95 0.95  0.5  0.1  0.1
## [3,] 0.50 0.50  1.0  0.5  0.5
## [4,] 0.10 0.10  0.5  1.0  0.9
## [5,] 0.10 0.10  0.5  0.9  1.0
```

```r
image(B, zlim=range(A))
title(main = 'Rank 4 approx')
B <- B + l[5] * matrix(v[,5],ncol=1) %*% matrix(v[,5],nrow=1)
round(B,2)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  1.0  0.9  0.5  0.1  0.1
## [2,]  0.9  1.0  0.5  0.1  0.1
## [3,]  0.5  0.5  1.0  0.5  0.5
## [4,]  0.1  0.1  0.5  1.0  0.9
## [5,]  0.1  0.1  0.5  0.9  1.0
```

```r
image(B, zlim=range(A))
title(main = 'Rank 5 approx')
par(op)
```

---

The eigendecomposition is also useful to find matrix inverses, and the power of a matrix. In fact, it is used to extend the notion of power of a matrix beyond integer powers:

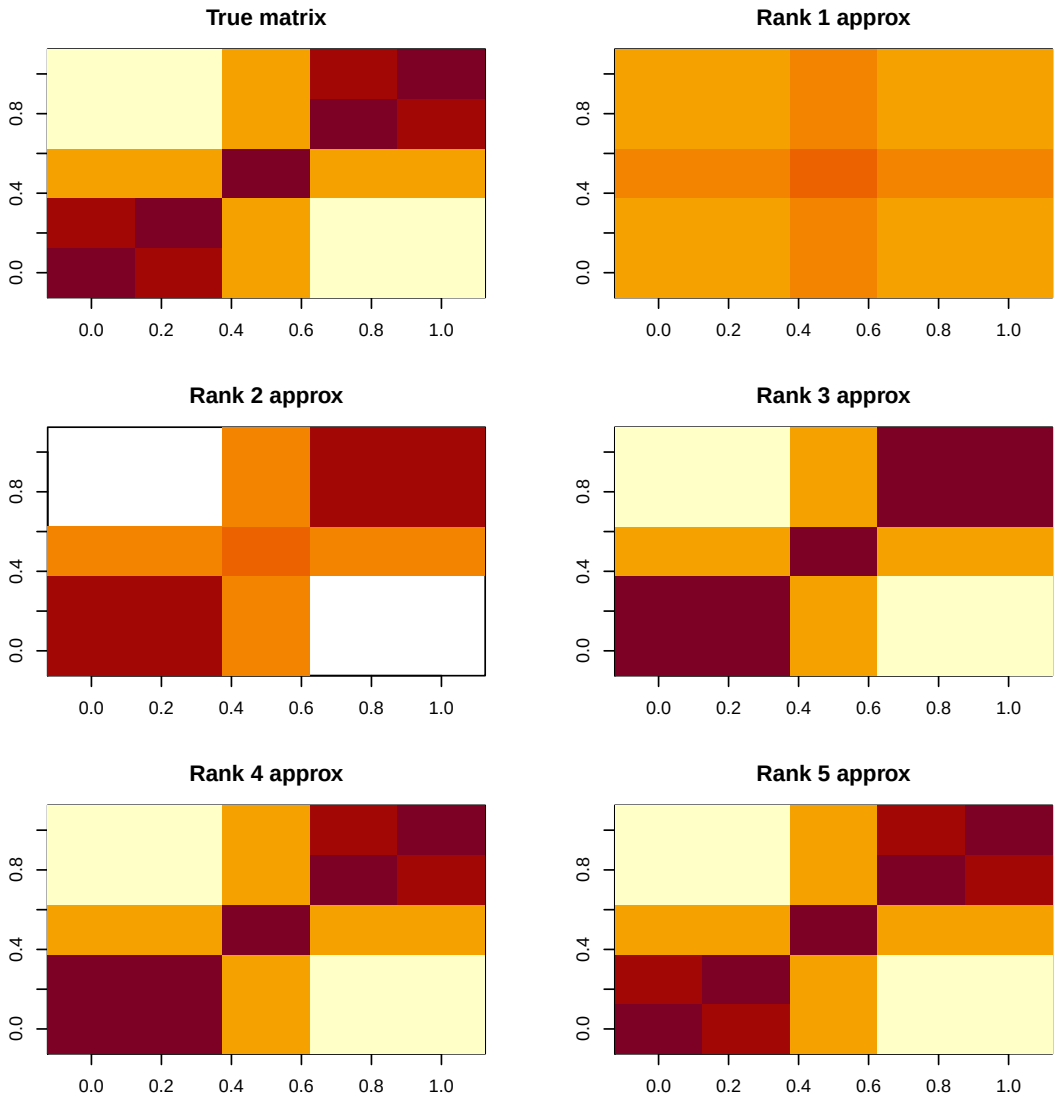



Figure 1: A matrix and its approximations using the eigendecomposition

**Definition 2.1.9 (Power of a symmetric matrix).** Let $A = E\Lambda E^\mathsf{T}$ be a symmetric positive semi-definite $p \times p$ matrix with eigenvalues in the diagonal matrix $\Lambda$ and orthonormal eigenvectors as columns in $E$. We define the $k$-th power of $A$ by

$$A^k = E\Lambda^k E^\mathsf{T}, \qquad k \in \mathbb{R}$$

where $\Lambda^k = \text{diag}(\lambda_1^k, \ldots, \lambda_p^k)$.

Verify that this definition indeed extends the intuitive definition of the power of a matrix.

## 2.2 Singular Value Decompositions, or SVD

**Theorem 2.2.1 (Singular value decomposition, or SVD).** Any $n \times p$ real matrix $X$ admits a decomposition of the form

$$X = ULV^\mathsf{T}, \tag{2.2.1}$$

where $U$ is an $n \times n$ orthogonal matrix, $L$ is a $n \times p$ rectangular diagonal matrix ($l_{ij} = 0$ for $i \neq j$) and $V$ is a $p \times p$ orthogonal matrix. The decomposition (2.2.1) is called a **singular value decomposition (SVD)** of $X$. The columns of $U$ are called left-singular vectors and the columns of $V$ right-singular vectors.

*Proof.* See video. □

The $l_{ii}$s are called the *singular values* of $X$, and they are usually ordered in a non-increasing order: $l_{11} \geq l_{22} \geq \cdots \geq l_{rr}$, where $r = \min(n, p)$. Sometimes we will write $l_i(X)$ for the largest $i$-th singular value of $X$. Simple algebra shows that the SVD of $X$ can be equivalently expressed as

$$X = \sum_{i=1}^{r} l_{ii} \mathbf{u}_i \mathbf{v}_i^\mathsf{T},$$

where $r = \text{rank}(X)$ and $\mathbf{u}_i$ and $\mathbf{v}_i$ are the $i^{th}$ columns in $U$ and $V$ respectively. This expression is a straightforward extension of the analogous result for the eigen-decomposition of a symmetric matrix, where the summation is truncated at $r = \text{rank}(X) \leq \min\{n, p\}$ since $l_{ii} = 0$ for $i > r$. By convention, $l_{11} \geq l_{22} \geq \ldots \geq 0$, so intuitively one may obtain a low-rank approximation to $X$ by taking the first few elements in the summation.

The following result gives a direct connection between the SVD of an $n \times p$ matrix $X$ and the eigendecompositions of $XX^\mathsf{T}$ and $X^\mathsf{T}X$.

**Proposition 2.2.2 (Link between spectral decomposition and SVD).** Let $X$ be an $n \times p$ real matrix.

1. If $X = ULV^\mathsf{T}$ is a SVD, then the columns of $U$ are eigenvectors of

$XX^\mathsf{T}$, the columns of $V$ are eigenvectors of $X^\mathsf{T}X$, and the corresponding eigenvalues are the square of the diagonal entries of $L$.

2. If $X^\mathsf{T}X = V\Lambda V^\mathsf{T}$ is an eigendecomposition, then one can give a SVD of the form $X = ULV^\mathsf{T}$, where $U$ contains as columns eigenvectors of $XX^\mathsf{T}$, and $L$ is diagonal with entries the square root of the diagonal entries of $\Lambda$.

This means more or less that (note that the following is **not** rigorous; try to understand why):

$$\left.\begin{aligned} X &= ULV^\mathsf{T} \\ XX^\mathsf{T} &= E\Lambda E^\mathsf{T} \\ X^\mathsf{T}X &= \widetilde{E}\widetilde{\Lambda}\widetilde{E}^\mathsf{T} \end{aligned}\right\} \Leftrightarrow \begin{cases} U = E \\ V = \widetilde{E} \\ (\Lambda)_{ii} = (\widetilde{\Lambda})_{ii} = (L)_{ii}^2. \end{cases}$$

*Proof.* Statement 1 follows from direct calculations. For the second statement, see proof of Theorem 2.2.1 □

Notice that by setting $Y = X^\mathsf{T}$ in point 2. of the Proposition, we get that given a spectral decomposition of $XX^\mathsf{T} = U\Lambda U^\mathsf{T}$ we can construct an SVD of the form $X = ULV^\mathsf{T}$.

The SVD is important, because its truncation provides the best low rank approximation of $X$. Let $\|X\|_F = \sqrt{\sum_{i,j}(X)_{ij}^2}$ denote the *Frobenius norm* of the matrix $X$.

**Theorem 2.2.3 (Eckart–Young–Mirsky Theorem).** Let $X$ be an $n \times p$ real matrix with SVD $X = ULV^\mathsf{T}$, and let

$$X_k = \sum_{i=1}^{k} l_i(X)\mathbf{u}_i\mathbf{v}_i^\mathsf{T}.$$

Then

$$\|X - X_k\|_F \leq \|X - A\|_F, \quad 1 \leq k \leq \min(n, p),$$

for all matrices $A$ with rank at most $k$.

*Proof.* See exercises. □

An example of application of the Eckart–Young–Mirsky Theorem is to image compression. An image is matrix of pixels, and each pixel is decomposed as a proportion of Red, Green, and Blue (RGB) colors. Thus an image can be viewed as 3 matrices of the same size. Conducting an SVD of each of the matrices separately, truncating the SVD (e.g. to get $X_k$ instead of $X$), and piecing the approximations back together to get an image produces a compressed version of the original image. Figure 2 shows an example of image compression for different truncation levels.

```
load("~/st323/data/pictures.rdata")
img <- images[[1]]
dims <- dim(img); m <- dims[1]; n <- dims[2]

decompositions <- list()
decompositions[[1]] <- svd(img[,,1])
decompositions[[2]] <- svd(img[,,2])
decompositions[[3]] <- svd(img[,,3])

p <- min(m,n)

# plot side-by-side the original image and the singular values
op <- par(mai=c(0,0,.2,0), oma=rep(0,4), mfrow=c(3,2))

viewImage(img, main='Original image')
for (k in c(1,4,10,50,100)) {
  approximation <- array(0,c(m,n,3))
  for (i in 1:3) {
    if (k == 1) {
      approximation[,,i] <- decompositions[[i]]$d[1]*
decompositions[[i]]$u[,1]%*%t(decompositions[[i]]$v[,1])
    } else {
      approximation[,,i] <- decompositions[[i]]$u[,1:k]%*%
diag(decompositions[[i]]$d[1:k])%*%t(decompositions[[i]]$v[,1:k])
    }
  }
  # rescale the approximation so the values of the image matrix are in [0,1]
  maxval <- max(approximation); minval <- min(approximation)
  compressedImage <- (approximation - minval)/(maxval - minval)

  # view the original and compressed image side-by-side
  viewImage(compressedImage, main=paste0("Rank ", k, " approximation"))
}
par(op)
```

## 2.3 Projection along Vectors, Angles, and Correlation

Let $\mathbf{x}$ and $\mathbf{y}$ be vectors of the same dimension. The projection of $\mathbf{y}$ onto $\mathbf{x}$ is defined to be

$$\mathbf{z} = \frac{\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{y}}{\mathbf{x}^\mathsf{T}\mathbf{x}} = \tilde{\mathbf{x}}\tilde{\mathbf{x}}^\mathsf{T}\mathbf{y},$$

where $\tilde{\mathbf{x}} = \mathbf{x}/\sqrt{\mathbf{x}^\mathsf{T}\mathbf{x}}$ is the normalized version of $\mathbf{x}$ having length 1. The length of the projection of $\mathbf{y}$ onto $\mathbf{x}$ is

$$\sqrt{\mathbf{z}^\mathsf{T}\mathbf{z}} = \sqrt{\frac{(\mathbf{y}^\mathsf{T}\mathbf{x})^2}{(\mathbf{x}^\mathsf{T}\mathbf{x})^2}\mathbf{x}^\mathsf{T}\mathbf{x}} = \frac{|\mathbf{y}^\mathsf{T}\mathbf{x}|}{\sqrt{\mathbf{x}^\mathsf{T}\mathbf{x}}} = |\mathbf{y}^\mathsf{T}\tilde{\mathbf{x}}|.$$

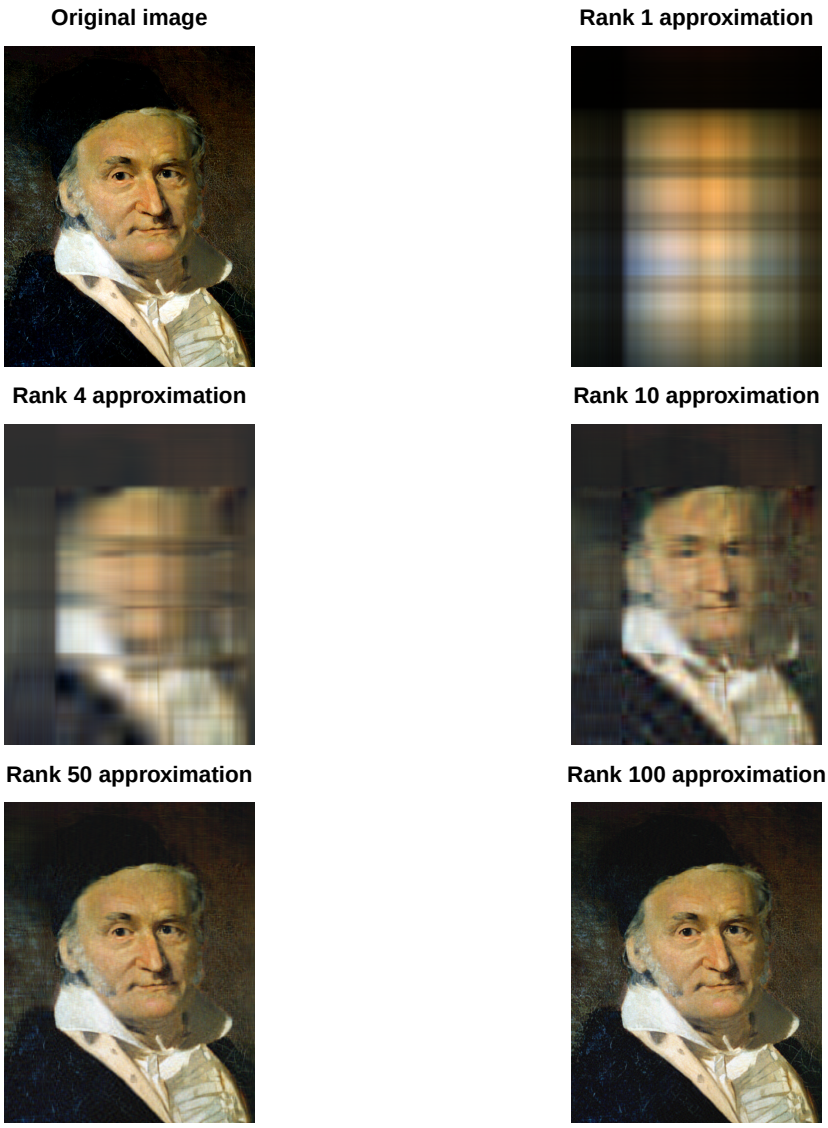



Figure 2: Compression of a picture of Carl Friedrich Gauss using SVD with different truncation levels (or low-rank approximation) of the SVD representation.

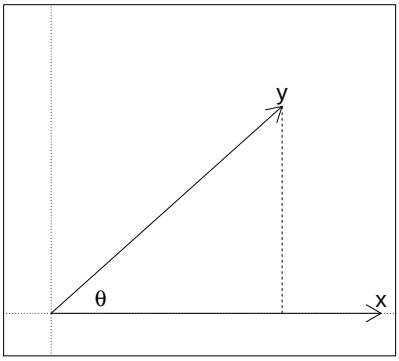



Figure 3: Angle $\theta$ between two vectors $\mathbf{x}$ and $\mathbf{y}$

Notice that the projection of $\mathbf{y}$ onto $\mathbf{x}$ depends on the inner-product (or dot product) $\tilde{\mathbf{x}}^\mathsf{T}\mathbf{y}$ of $\tilde{\mathbf{x}}$ and $\mathbf{y}$, sometimes also written $\langle\tilde{\mathbf{x}}, \mathbf{y}\rangle$. The inner-product $\tilde{\mathbf{x}}^\mathsf{T}\mathbf{y}$ can be thought of as the signed length of $\mathbf{z}$.

If we have a matrix $A$ where each row has unit length, then the vector $\mathbf{z} = A\mathbf{y}$ gives the inner-products (or signed lengths) of the projections of $\mathbf{y}$ onto each row in $A$. Indeed, notice that $A$ is simply a series of row vectors stacked on top of each other, hence each element in $\mathbf{z}$ represents the inner-product of $\mathbf{y}$ and the vector stored in the corresponding row in $A$.

An interesting notion closely related to the inner-product between vectors is the *angle between two vectors*.

**Definition 2.3.1 (Angle between vectors).** Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors in $\mathbb{R}^p$. The angle between $\mathbf{x}$ and $\mathbf{y}$ is the value $\theta$ such that $\cos(\theta) = \frac{\mathbf{x}^\mathsf{T}\mathbf{y}}{\sqrt{\mathbf{x}^\mathsf{T}\mathbf{x}}\sqrt{\mathbf{y}^\mathsf{T}\mathbf{y}}}$.

Let us verify that this definition makes sense. Notice that this definition is invariant to rotations and scalings, i.e. the angle between $\lambda R\mathbf{x}$ and $\gamma R\mathbf{y}$ is the same as the angle between $\mathbf{x}$ and $\mathbf{y}$, where $\lambda, \gamma > 0$ and $R$ is a rotation matrix (i.e. $R^\mathsf{T}R = I, \det(R) = 1$). We can therefore assume that $\mathbf{x} = (1, 0, \ldots, 0)^\mathsf{T}$ and $\mathbf{y} = (y_1, y_2, 0, \ldots, 0)^\mathsf{T}$ such that $y_1^2 + y_2^2 = 1$. Then working in the plane spanned by $(1, 0, \ldots, 0)$ and $(0, 1, 0, \ldots, 0)$, we can see that the angle $\phi$ between $\mathbf{x}$ and $\mathbf{y}$ satisfies $\cos(\phi) = y_1 = \mathbf{x}^\mathsf{T}\mathbf{y}/\sqrt{(\mathbf{x}^\mathsf{T}\mathbf{x})(\mathbf{y}^\mathsf{T}\mathbf{y})}$. Therefore definition 2.3.1 makes sense. Figure 3 illustrates the notion of angle between two vectors.

This notion of angle between vectors is particularly interesting to statisticians. Suppose now that $\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$ and $\mathbf{y} = (y_1, \ldots, y_n)^\mathsf{T}$ are the observed values for two random variables, and suppose that we have centered these values so that the sample means are 0. Then the cosine of the angle between them is given by

$$\frac{\mathbf{x}^\mathsf{T}\mathbf{y}}{\sqrt{\mathbf{x}^\mathsf{T}\mathbf{x}}\sqrt{\mathbf{y}^\mathsf{T}\mathbf{y}}} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2}\sqrt{\sum_{i=1}^n y_i^2}} = \frac{\frac{1}{n-1}\sum_{i=1}^n x_i y_i}{s_x s_y} = r_{xy}, \tag{2.3.1}$$

where $s_x, s_y$ are the sample standard deviations and $r_{xy}$ is the Pearson correlation coefficient. **Hence the Pearson correlation has a geometric interpretation as the cosine of the angle between the two vectors.** Throughout this course we will see other cases in which there is a duality between algebra and geometry, and that this can help us interpret statistical procedures.

## 2.4 Orthogonal Projectors

Later in the module we will be interested in projecting data in $\mathbb{R}^p$ into either a small dimensional space $\mathbb{R}^d$, or into a lower dimensional subspace of $\mathbb{R}^p$. There are various ways of projecting onto a subspace $V \subset \mathbb{R}^p$, but the most useful for us will be the orthogonal projection, that is the mapping $P : \mathbb{R}^p \to \mathbb{R}^p$ such that $\langle P\mathbf{x}, \mathbf{y} - P\mathbf{y} \rangle = 0$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$. Writing $P$ as a matrix, it can be shown that $P = P^\mathsf{T} = P^2$, and we use this as a definition of orthogonal projection:

**Definition 2.4.1 (Orthogonal Projector).** A $p \times p$ matrix $P$ is an **orthogonal projector** if $P^\mathsf{T} = P = P^2$. The corresponding mapping $\mathbb{R}^p \to \mathbb{R}^p$ is called an **orthogonal projection**.

Notice that if $B$ is $p \times q$ with $B^\mathsf{T}B = I_q$, then $BB^\mathsf{T}$ is an orthogonal projection matrix. The following proposition tells us that every orthogonal projection matrix is of this form:

**Proposition 2.4.2 (Characterization of Orthogonal Projectors).** Let $P$ be a $p \times p$ orthogonal projector. Then

1. The eigenvalues of $P$ are either 0 or 1,
2. $q = \mathrm{Tr}(P) \in \{0, 1, \ldots, p\}$,
3. $P = BB^\mathsf{T}$ for some $p \times q$ matrix $B$ satisfying $B^\mathsf{T}B = I_q$.

*Proof.* See video. □

The following result, known sometimes as *Poincaré's inequalities*, will be useful later in the Module.

**Lemma 2.4.3 (Poincaré's inequalities).** For any $p \times p$ orthogonal projection matrix $P$,

$$\lambda_{p-q+1}(A) + \cdots + \lambda_p(A) \leq \mathrm{Tr}(AP) \leq \lambda_1(A) + \cdots + \lambda_q(A),$$

for all $p \times p$ symmetric matrices $A$, where $q = \mathrm{Tr}(P)$ and $\lambda_i(A)$ is the $i$-th largest eigenvalue of $A$.

*Proof.* See exercises. □

## 2.5 Measuring distances

Many multivariate methods are based on measuring distances between objects in an appropriate manner. How we measure distance does matter, for instance the distances between cities in the UK measured in miles can be quite different from the distances measured in time required to travel between them. Two cities far away in space could have airports and hence be close in terms of traveling time. Of course, how to measure distances adequately is application-specific, but there are some default choices that can be helpful in many scenarios.

**Definition 2.5.1 (Distance or Metric).** We say that a function $d(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, assigning a real value $d(\mathbf{x},\mathbf{y}) \in \mathbb{R}$ to each pair $(\mathbf{x},\mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^p$ is a **distance** (or **metric**) if it satisfies the following properties: $\forall \mathbf{x},\mathbf{y} \in \mathbb{R}^p$,

1. $d(\mathbf{x},\mathbf{y}) \geq 0$, and $d(\mathbf{x},\mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$. (**positivity**)
2. $d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x})$ (**symmetry**).
3. $d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{z}) + d(\mathbf{z},\mathbf{y})$, $\forall \mathbf{z} \in \mathbb{R}^p$ (**triangle inequality**).

Here are some possible distances:

- Euclidean distance: $d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2}$. The points satisfying $d(\mathbf{x},\mathbf{y}) = c$ lie on a circle. A limitation is that it does not consider that the elements in $\mathbf{x}$ might be on different scales (*e.g.* if $x_1$ is in millimeters and $x_2$ on kilometers then $x_1$ will tend to have a higher weight on the Euclidean distance).
- Scaled Euclidean distance: $d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2/s_i}$, where $s_i > 0$ is a scaling constant such as the pooled variance of $x_i$ and $y_i$. The points satisfying $d(\mathbf{x},\mathbf{y}) = c$ lie on an ellipse with main axes on the canonical basis. This distance accounts for differences in scale, but does not consider that the components in $\mathbf{x}$ could be highly correlated.
- Mahalanobis distance: $d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\mathsf{T} S^{-1} (\mathbf{x}-\mathbf{y})}$, where $S$ is a $p \times p$ PSD matrix. The points satisfying $d(\mathbf{x},\mathbf{y}) = c$ lie on an ellipse with general axes. The Mahalanobis distance sets $S$ to be the covariance matrix (assumed to be equal for $\mathbf{x}$ and $\mathbf{y}$). In that case, $\mathbf{z}_x = S^{-1/2}\mathbf{x}$ can be seen as an uncorrelated version of $\mathbf{x}$ (similarly $\mathbf{z}_y = S^{-1/2}\mathbf{y}$), and the Mahalanobis distance is simply the Euclidean distance between $\mathbf{z}_x$ and $\mathbf{z}_y$. Let us see why.

$$(\mathbf{x}-\mathbf{y})^\mathsf{T} S^{-1/2} S^{-1/2} (\mathbf{x}-\mathbf{y}) = (S^{-1/2}\mathbf{x} - S^{-1/2}\mathbf{y})^\mathsf{T} (S^{-1/2}\mathbf{x} - S^{-1/2})\mathbf{y} = (\mathbf{z}_x - \mathbf{z}_y)^\mathsf{T} (\mathbf{z}_x - \mathbf{z}_y) = \sum_{i=1}^p (z_{xi} - z_{yi})^2$$

The Mahalanobis distance is based on the idea of quadratic forms.

**Definition 2.5.2 (Quadratic form).** A quadratic form in the $p$ variables $\mathbf{x} = (x_1, \ldots, x_p)^\mathsf{T}$ is a function $Q(\mathbf{x}) = \mathbf{x}^\mathsf{T} A \mathbf{x}$, where $A$ is a $p \times p$ symmetric matrix.

An equivalent way to write the quadratic form is $Q(\mathbf{x}) = \sum_{i=1}^{p} a_{ii} x_i^2 + 2 \sum_{i<j} a_{ij} x_i x_j$, which shows that a quadratic form is quite simple: it only contains squared terms and cross-product terms, with coefficients given by the elements in $A$.

# 3 Basic Exploratory Data Analysis

When you acquire a new dataset, you need to first get a feel for it, before turning to a formal statistical analysis. This is called *Exploratory Data Analysis*, or EDA, and essentially it means that you play with your data. In this section, we explore basic EDA. Throughout the module, we will illustrate how the new concepts can be used for EDA.

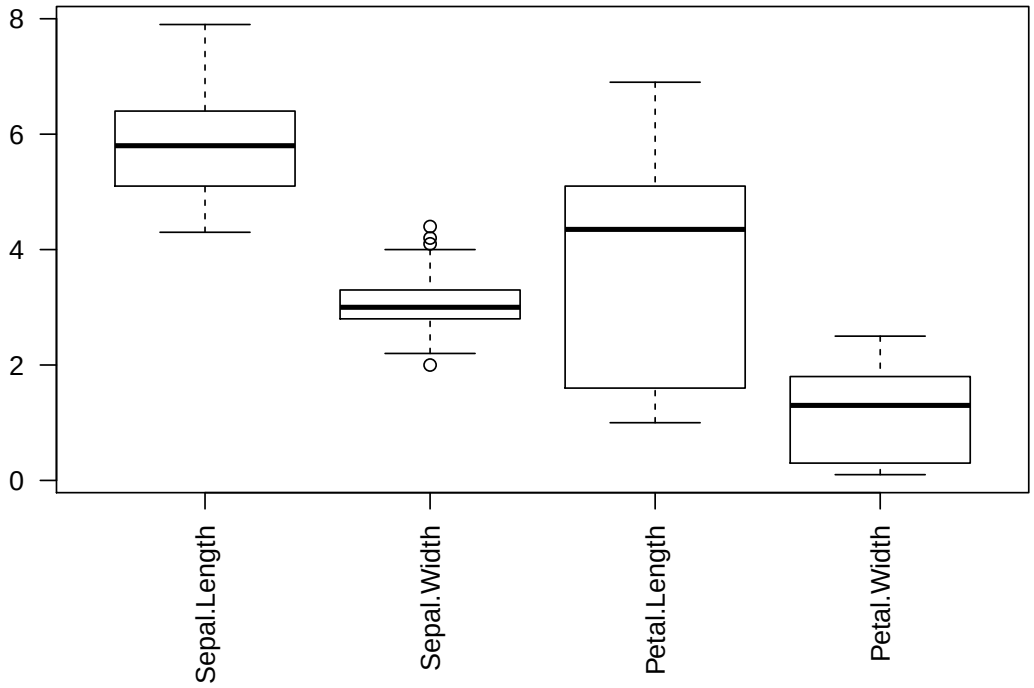
## 3.1 Sanity Checks and Univariate Plots

The first things to do with your data is to get some basic summaries of the data, and understand what each variables are.

```
#?iris
str(iris)

## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1

summary(iris)

##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

We see that the first 4 variables represent a length/width. Therefore these need to be positive. Inspection of the minimum value for each variable confirms that there is no problem with this. The same can be seen via boxplots:

```
op = par(las=2, mar=c(6,4,4,1))
boxplot(iris[,-5], horiz=T)
par(op)
```




However, the boxplots tell us a bit more. In particular, it seems that Petal < Sepal usually. See Figure 4 to recall what a petal and a sepal are. Now the data seems a bit odd... until we see a picture of an iris setosa (see Figure 5).


Figure 4: Flower parts (taken from Wikipedia)

You can go further and look at histograms of you data, for instance:

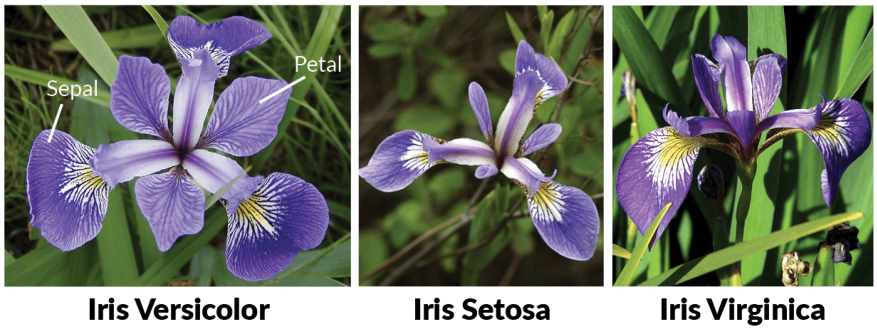
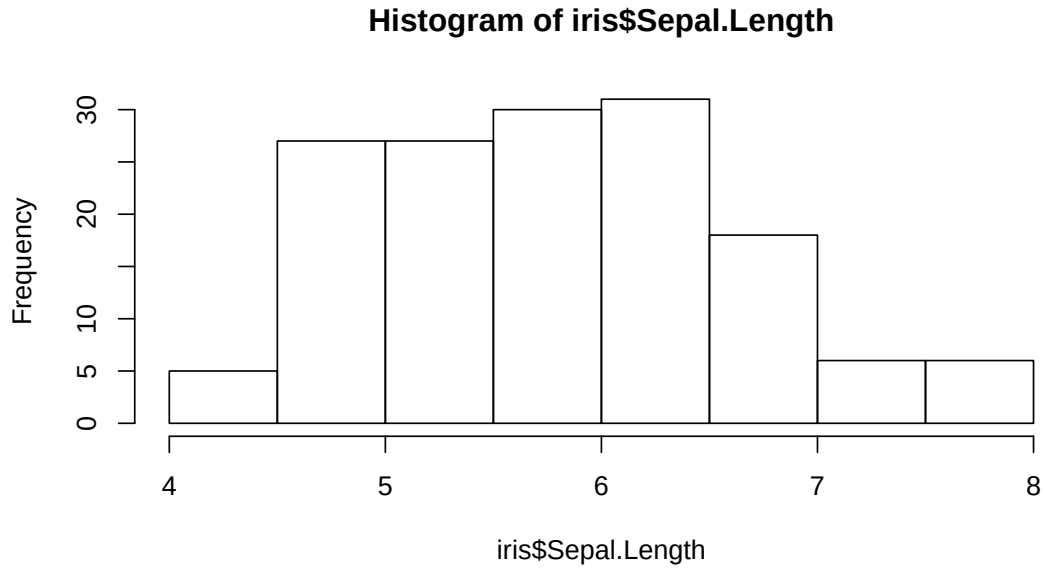



Figure 5: Iris parts (ref: `http://www.lac.inpe.br/~rafael.santos/Docs/R/CAP394/WholeStory-Iris.html`)

```
hist(iris$Sepal.Length)
```




Since the data is multidimensional, we now turn to more complex visualization methods. Visualizing high-dimensional data is challenging, not only for the impossibility of plotting more than 3-4 dimensions, but also because of our limitation as humans to perceive and understand large amounts of information. Here we discuss some basic plots that can help to get a first impression of what is going in the data. Later in the course we shall see more advanced techniques.

## 3.2 Scatterplots

Perhaps the easiest approach is to produce a matrix of bivariate scatterplots. While these are not truly multivariate plots, they can still be quite helpful in practice. Assume that your sample is $X_1, \dots X_n \in \mathbb{R}^p$, for instance here $X_1 = (x_{11}, \dots, x_{14})$ is the vector of width and length of sepal and petal of flower 1 (we omit the name of the species here). You create a new plot consisting of a $4 \times 4$ grid of plots. In plot

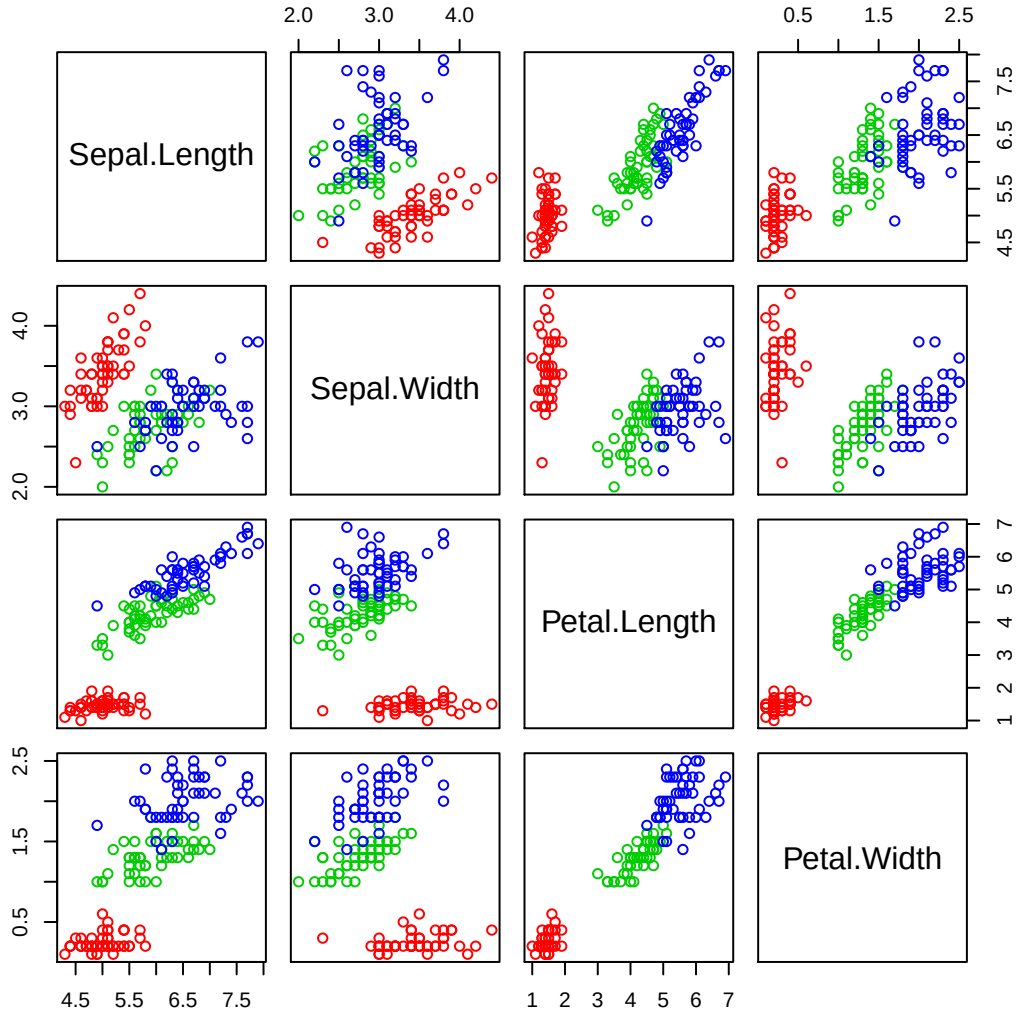



Figure 6: Anderson's iris data. Colors indicate the 3 species

$(i, j)$, you will draw the $n$ points $(x_{li}, x_{lj})_{l=1,\ldots,n}$, one for each observation in your sample. If you have an additional categorical variable for each observation (here the iris species), you can color-code it and color each point accordingly.

```
col <- 1+as.numeric(factor(iris[,5]))
plot(iris[,1:4],col=col)
#pairs(iris[,1:4],col=col) ## gives the same result
```

Figure 6 shows an example for Anderson's iris data, which contains 4 continuous variables and a categorical variable indicating the flower species. Each individual scatterplot reveals information regarding 3 variables (the two axis plus the group labels), and reveals some of the structure in the data.

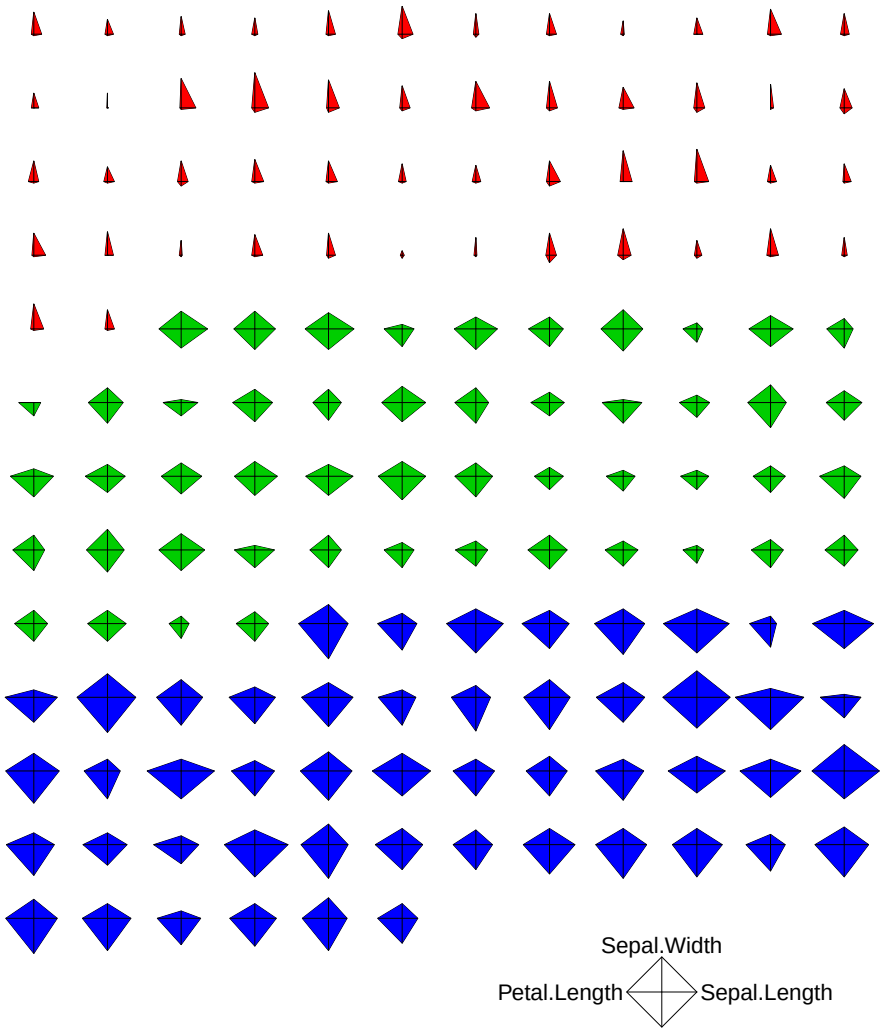



Figure 7: Star plot of the Iris dataset, colored by species

## 3.3 Star plots

Star plots display, for each individual, segments emanating from a central point. The length of each segment indicates the value of an individual variable.

```
require(grDevices)
stars(iris[,-5], col.stars=col, key.loc = c(20, 0))
```

We see clearly in Figure 7 the difference between the "red" species and the other species.

Let's see another example with a dataset measuring 7 continuous (horse power, miles per gallon etc.) and 4 discrete variables for 32 different cars. This dataset is called `mtcars`, and Figure 8 shows the related star plots.

```r
str(mtcars)
```

```
## 'data.frame': 32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```r
summary(mtcars)
```

```
##       mpg             cyl             disp             hp             drat      
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930  
##        wt             qsec             vs              am              gear      
##  Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000  
##  1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000  
##  Median :3.325   Median :17.71   Median :0.0000   Median :0.0000   Median :4.000  
##  Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688  
##  3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000  
##  Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000  
##       carb      
##  Min.   :1.000  
##  1st Qu.:2.000  
##  Median :2.000  
##  Mean   :2.812  
##  3rd Qu.:4.000  
##  Max.   :8.000
```

```r
stars(mtcars[, 1:7], key.loc = c(14, 2), main = "Motor Trend Cars :
stars(*, full = F)", full = FALSE)
```

An obvious limitation with star plots is that when either the number of individuals or the number of variables grows the plot gets cluttered and hard to interpred. Another limitation is that the order in which we decide to plot the variables can change the final appearance of the plot.

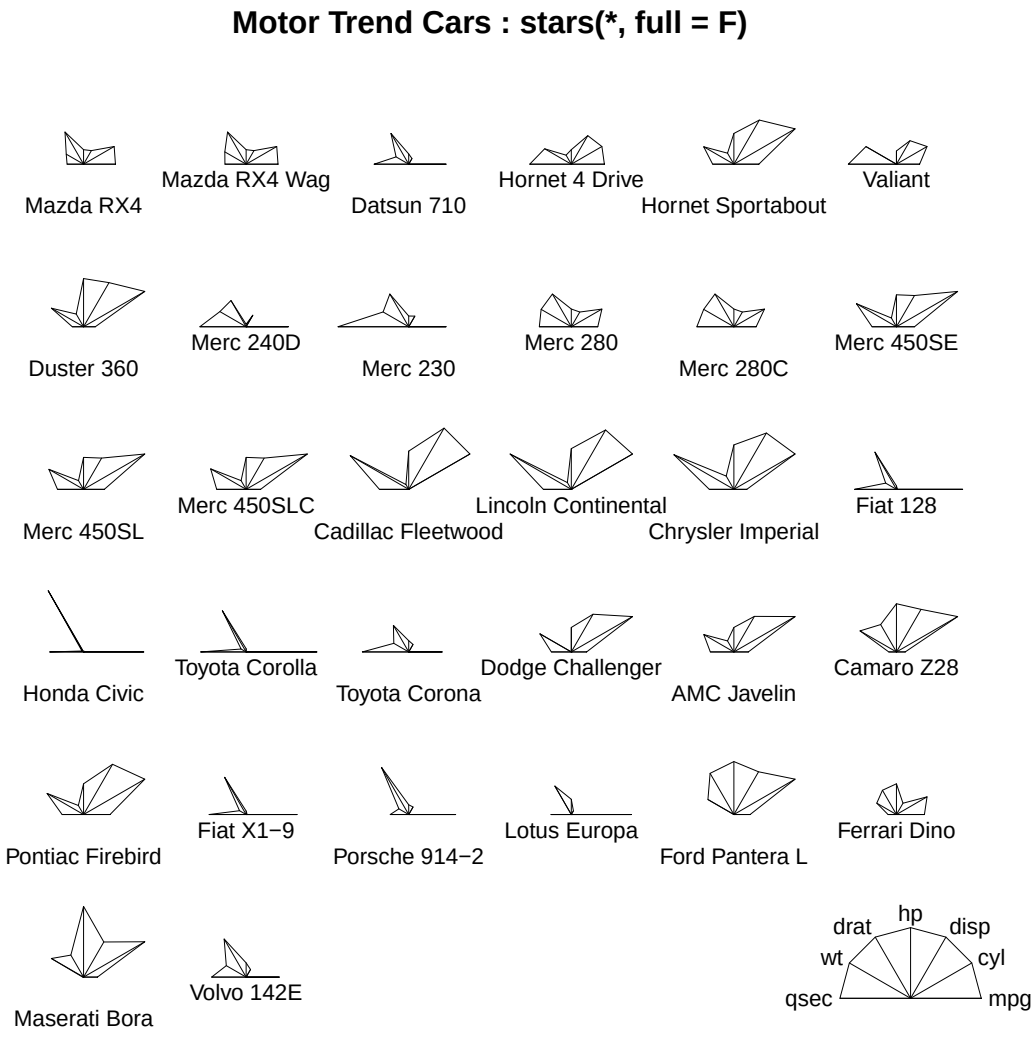



Figure 8: Star plot for `mtcars` dataset

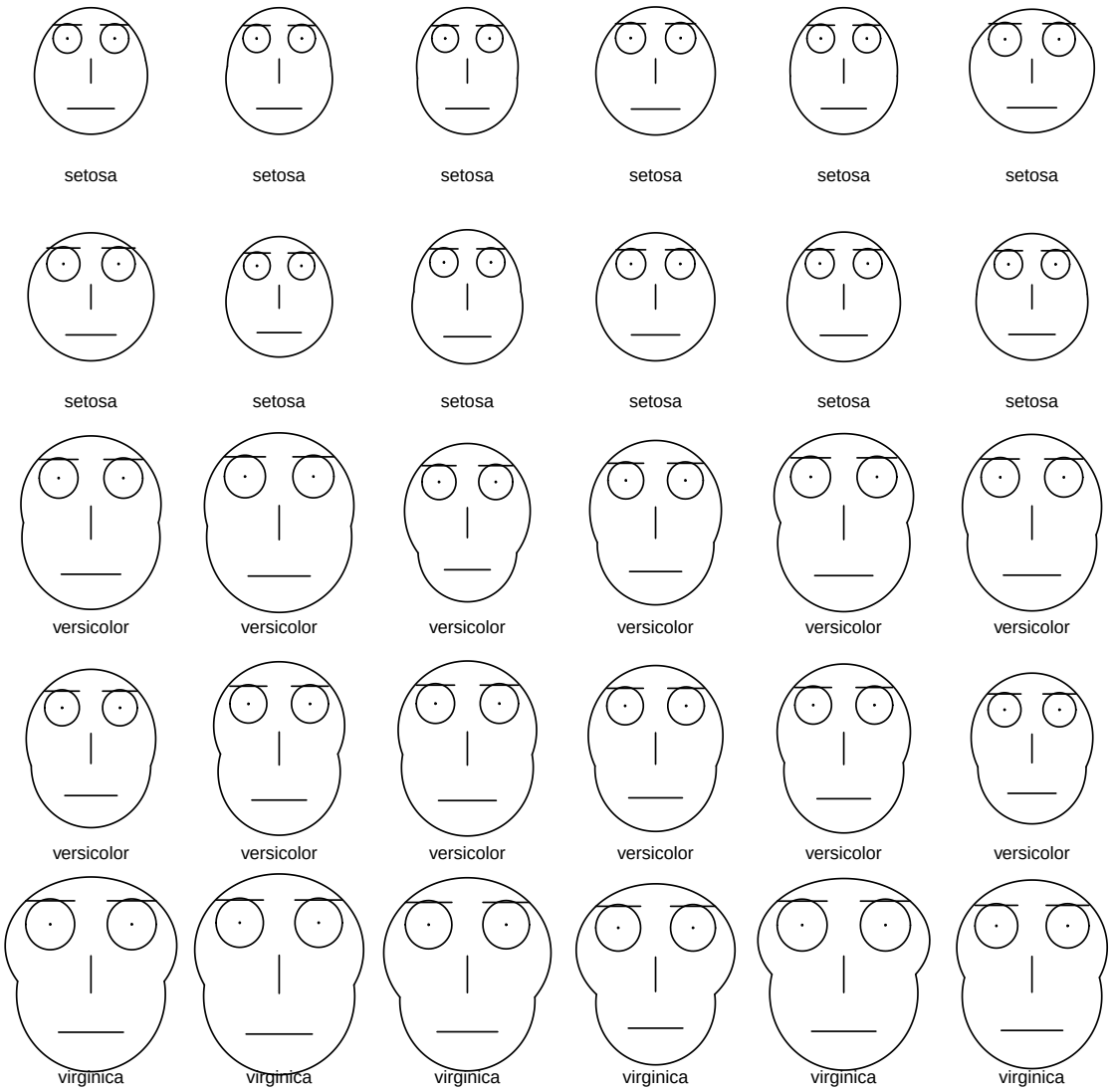



Figure 9: Chernoff faces for a subsample of the `iris` dataset

## 3.4 Chernoff faces

Chernoff faces are a fun alternative to starplots. They too plot each individual separately, but here each characteristic of the face is associated to a separate variable, and its size indicates its value.

```
library(TeachingDemos)
ii = sort(sample(nrow(iris), 30))
faces2(iris[ii,-5], labels=iris[ii,5])
```

```
library(TeachingDemos)
faces2(mtcars[,1:7])
```

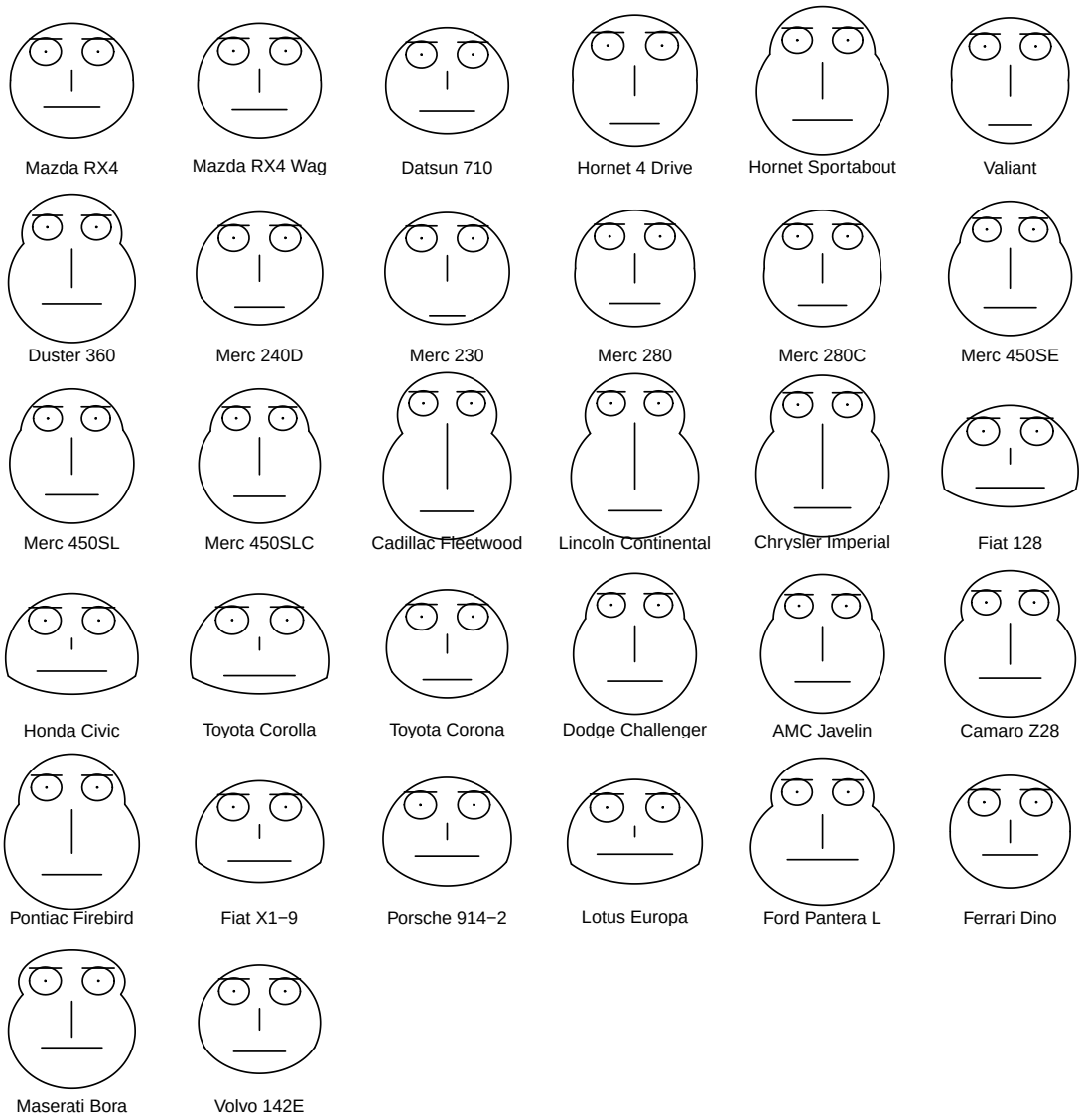



Figure 10: Chernoff faces for the `mtcars` dataset, the features are: 1 Width of center 2 Top vs. Bottom width (height of split) 3 Height of Face 4 Width of top half of face 5 Width of bottom half of face 6 Length of Nose 7 Height of Mouth.

For Figure 10, the value of the first variable is reflected in the width of the center of the face. The value of the second variable in the top vs. bottom width, the third variable in the height of the face etc.

As with star plots, Chernoff faces get messy when presenting more than a few individuals or variables, and the plot can look quite different depending on how we order the variables. In spite of these limitations, Chernoff faces are an example of using pictograms to represent multivariate data in an amenable dataset. For instance, in order to represent several characteristics of fruit (colour, size, weight) one could plot fruit pictures which reflect the desired characteristics. As another example, one could display basic medical data such as height, weight, gender and blood pressure using pictures of humans. These plots are sometimes seen in posters presented in scientific conferences, as they help attract the attention of the audience.

As a historical note, Chernoff faces were invented by Herman Chernoff, a mathematician, statistician and physicist who was at MIT and Harvard. He is rumoured to lament that, in spite of his many serious research contributions, he is most famous for his faces plot.

## 3.5 Andrews curves

The idea behind Andrews curves is to represent each individual as a function (or curve), such that each curve uniquely represents the value of a set of variables $\mathbf{x} = (x_1, \ldots, x_p)^\mathsf{T}$. The Andrews curve is obtained by calculating the function

$$f_\mathbf{x}(t) = x_1/\sqrt{2} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \cdots + x_p e_p(t),$$

and plotting this function over the range $t \in [-\pi, \pi]$, where $e_p(t) = \cos((p-1)t/2)$ if $p$ is odd, and $e_p(t) = \sin(pt/2)$ if $p$ is even. In this manner, each observation will appear as a separate curve in the plot.

Those mathematically inclined will recognize that the curve simply corresponds to a discrete Fourier transform. Some important properties of Andrews curves are:

1. There is a one-to-one correspondence between $\mathbf{x}$ and $f_\mathbf{x}(t)$. As a consequence, the curves for $\mathbf{x}_1$ and $\mathbf{x}_2$ will be equal if and only if $\mathbf{x}_1 = \mathbf{x}_2$.

2. The Euclidean distance between two observations $\mathbf{x}_1, \mathbf{x}_2$ is directly proportional to the distance between their corresponding curves, which is measured as
$$\sqrt{\int_{-\pi}^{\pi} (f_{\mathbf{x}_1}(t) - f_{\mathbf{x}_2}(t))^2 dt}.$$
That is, whenever two curves are similar for all $t$ then their corresponding $\mathbf{x}$ values are also similar to each other.

3. The functions reflect the variance in the data. When the variables are uncorrelated with common variance $\sigma^2$ and the number of variables $p$ is odd, then the variance of the functions at $t$ is directly proportional to $\sigma^2$ for any value of $t$. When $p$ is even, the variance depends slightly on $t$ but always lies between $\frac{1}{2}\sigma^2(p-1)$ and $\frac{1}{2}\sigma^2(p+1)$. In practice, what this property means is that by

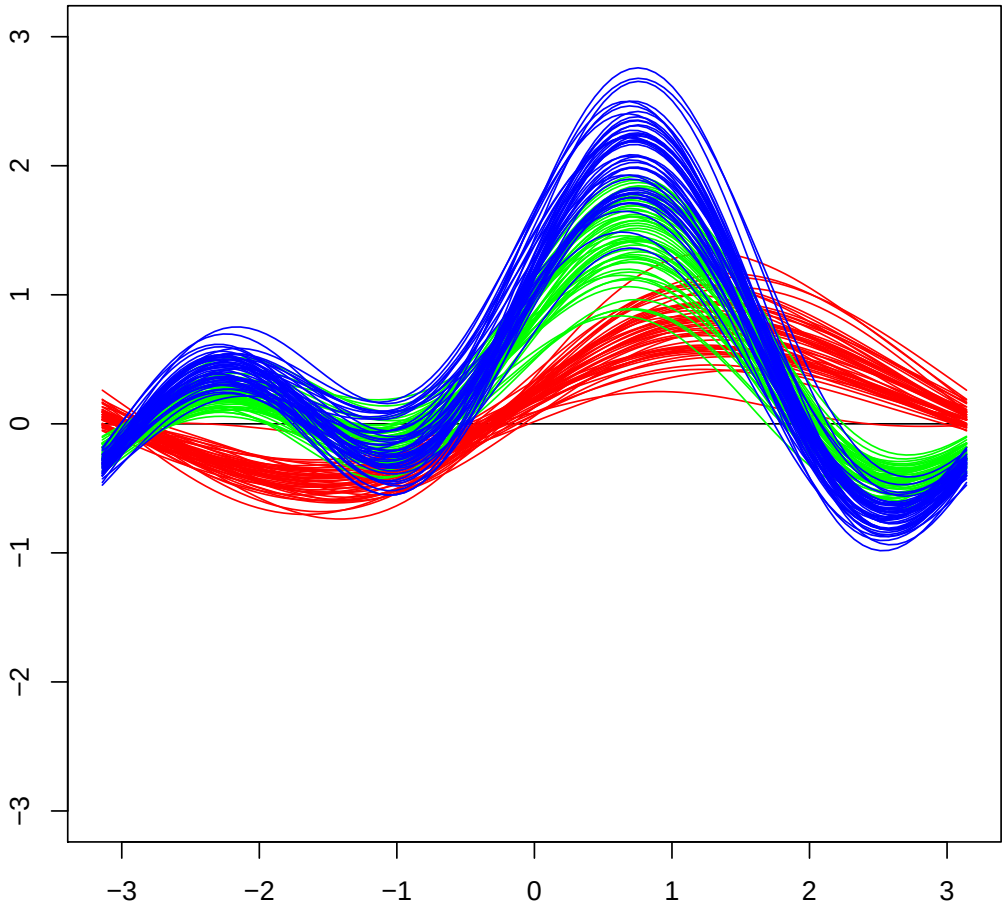



Figure 11: Andrews curves for Anderson's `iris` dataset. Each curve corresponds to a different individual, and colors indicate the 3 species.

looking at the variability of the functions for any value of $t$ we can get an idea of the variance of the variables in $\mathbf{x}$.

```
library(andrews)
andrews(iris[,1:5], clr=5, ymax=3)
```

Notice that in Figure 11, we can see that the three species are separated in terms of their $\mathbf{x}$ values. In fact, the species shown in red shows a clear separation (this was also apparent in the scatterplots in Figure 6), and seems to have lower overall variability, indicating that the individuals from this species are quite similar to each other.

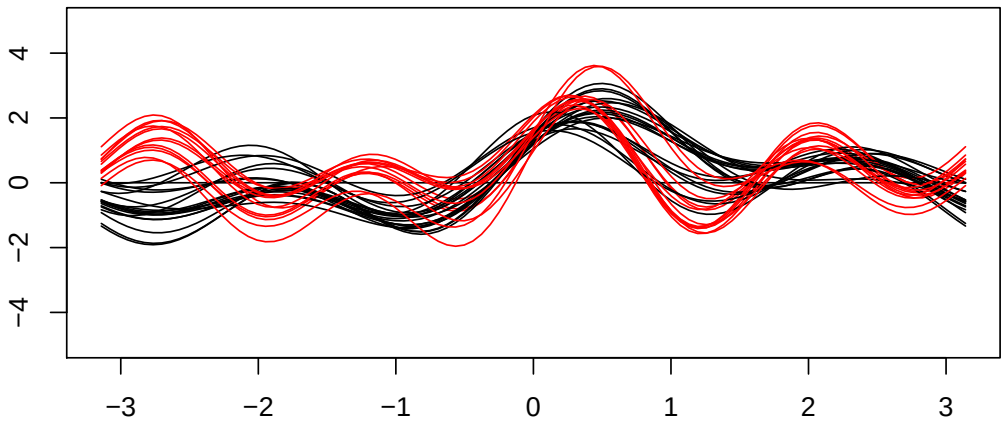



Figure 12: Andrews curves for `mtcars` dataset. Each curve corresponds to one car. The color indicates whether the car is an automatic or not.

```
andrews(mtcars[,c(1:7, 9)], clr=8, ymax=5)
```

In Figure 12, we see a clear difference between automatic and manual transmission cars at the left of the curves.

## 3.6 High-dimensional datasets

For high-dimensional datasets, we cannot plot pairs plots, Chernoff faces, or Andrews curves. As an example, let's try to do a pairs plot for 7 variables. The result is in Figure 13. We are still able to see something, but we are at the limit. If we had 20 variables, we'd get a plot like Figure 14. Now we really see nothing...

Let us give an example of a high-dimensional dataset. The dataset `ZIP code`, available at `https://web.stanford.edu/~hastie/ElemStatLearn/`, contains images ($16 \times 16$ pixels) of handwritten digits, along with a label telling us which digit this corresponds to. Here is a quick look at the data structure:

```
library(magrittr)
zip_code  <- read.table("~/st323/data/zip.train")
# download at https://web.stanford.edu/~hastie/ElemStatLearn/ or on the Moodle
is.data.frame(zip_code)

## [1] TRUE

dim(zip_code)

## [1] 7291  257
```

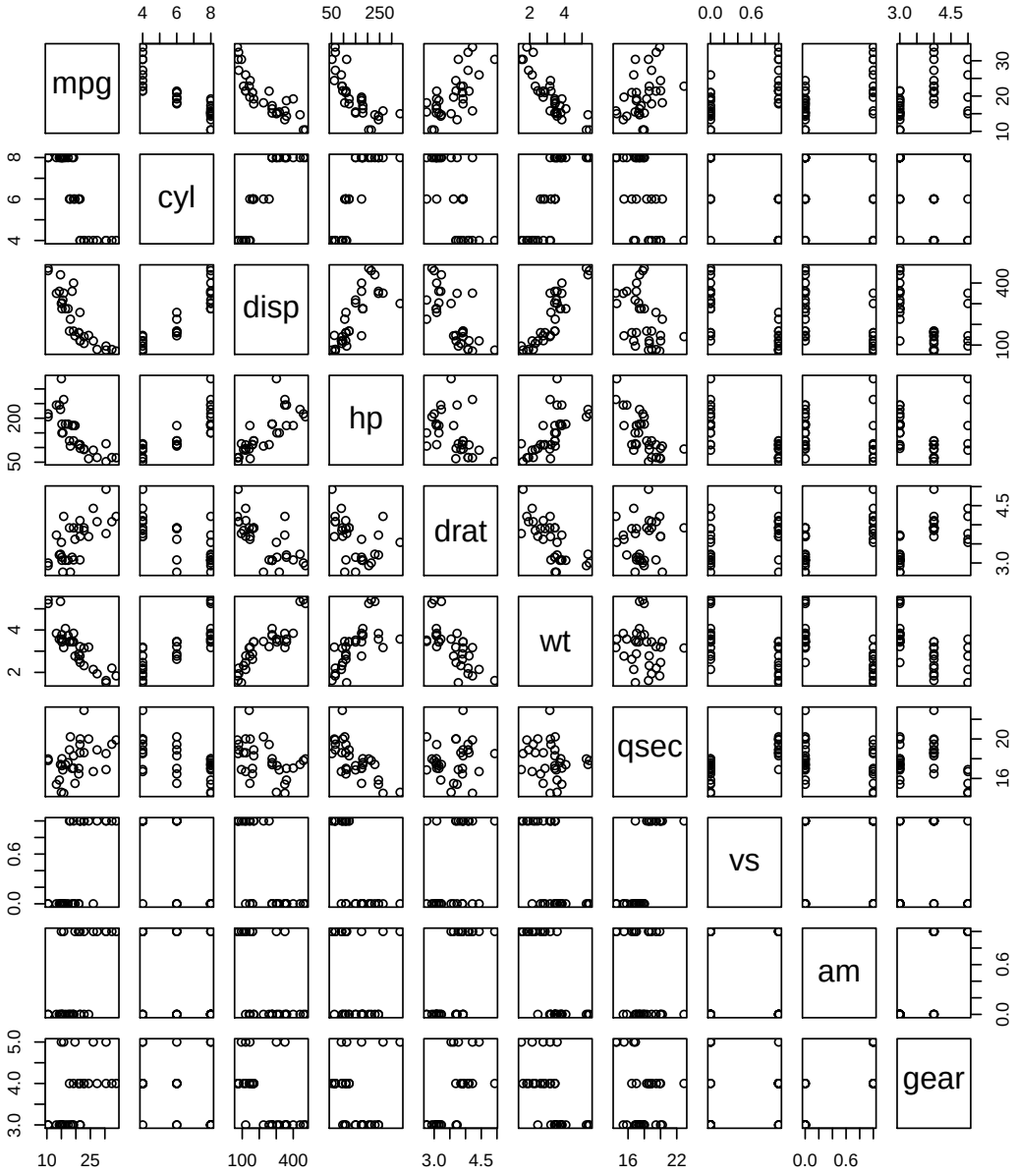



Figure 13: Pairs plot for 10 variables of `mtcars`

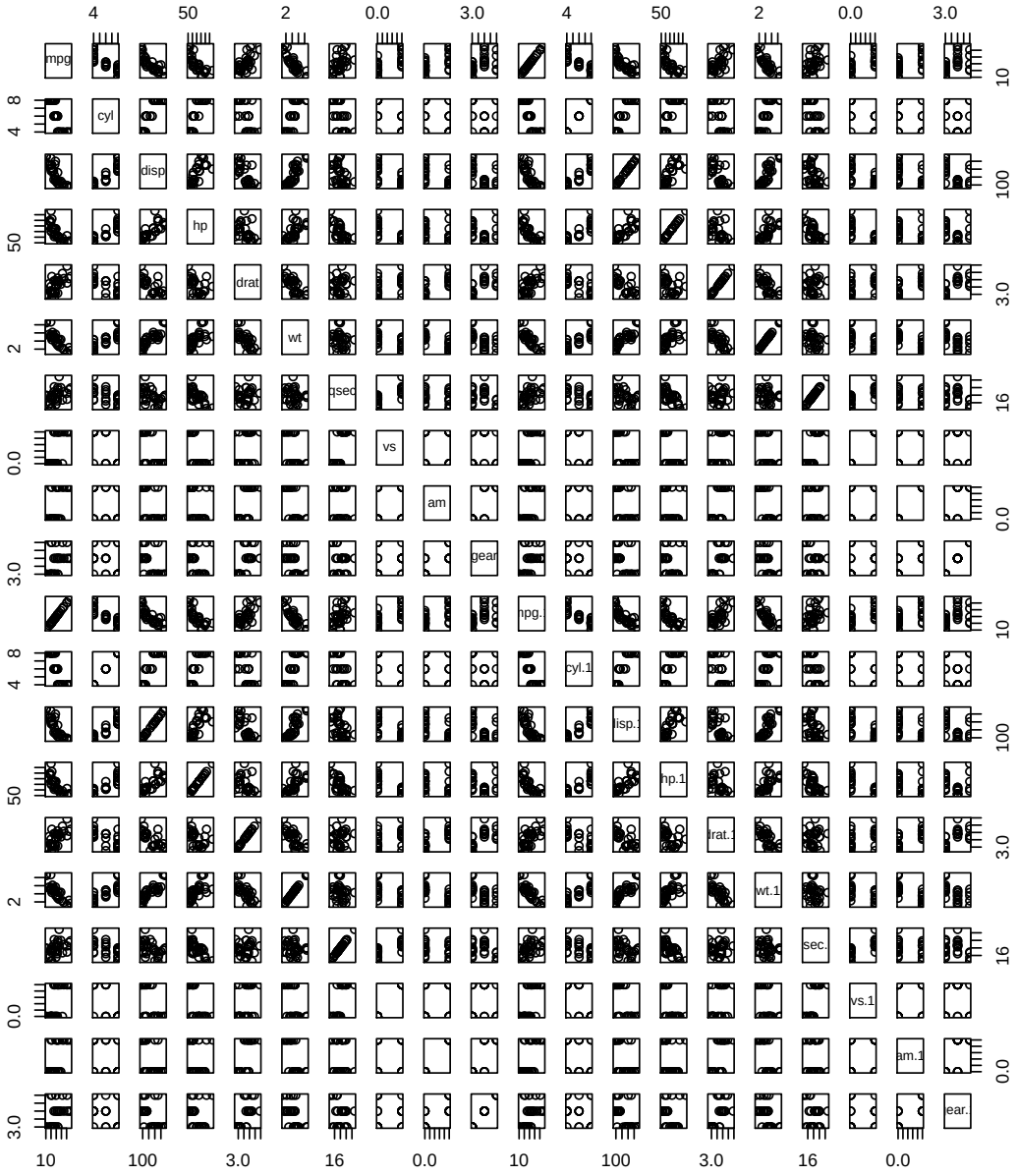



Figure 14: Pairs plot for an artificially augmented dataset with 20 variables

```
zip_code[,1] %>% head ## the first column contains the digit label

## [1] 6 5 4 7 3 6

zip_code[1,-1] %>% unlist %>% str ## the other columns contain the pixels

##  Named num [1:256] -1 -1 -1 -1 -1 -1 -1 -0.631 0.862 -0.167 ...
##  - attr(*, "names")= chr [1:256] "V2" "V3" "V4" "V5" ...

digit <- zip_code[,1] ## the digit label
```

Each row of `zip_code` contains the digit label, and the pixels of the corresponding image, which we've plotted in Figure 15.

```
    op <- par(mfrow=c(1,2))
    ## the pixels must be sorted as a matrix before being plotted
    zip_code[1,-1] %>% unlist %>% matrix(16,16) %>% image(col=grey(seq(1,0,
 len=100)), xaxt='n', yaxt='n')
    ## some care is needed to get the image in the correct orientation!
    #
    ## a simple function to plot the image of the digit
    ## input: x, the pixel values as a vector
    plot.digit = function(x, num=NULL, ...){
        x_image <-  x %>% matrix(16,16) %>% ( function(y){ y[, ncol(y):1]} )
        image(x_image, col=gray(seq(1,0, len=100)))
    }
    plot.digit(unlist(zip_code[1,-1]), xaxt='n', yaxt='n')
    par(op)
```

We now plot in Figure 16 one image of each digits $0, 1, \ldots, 8$ (we've omitted 9 for convenience).

```
##

    op <- par(mfrow=c(3,3), mai=rep(0.01,4))
    for( i in 0:8 ){
        j <- which(digit == i)[1]
        zip_code[j,-1] %>% unlist %>% plot.digit(xaxt='n', yaxt='n')
    }
    par(op)
```

Notice that our data is 256 dimensional! Forget about pair plots. Can we do Chernoff faces? What about Andrew's curves?

```
faces2(zip_code[1:40, -1])
```

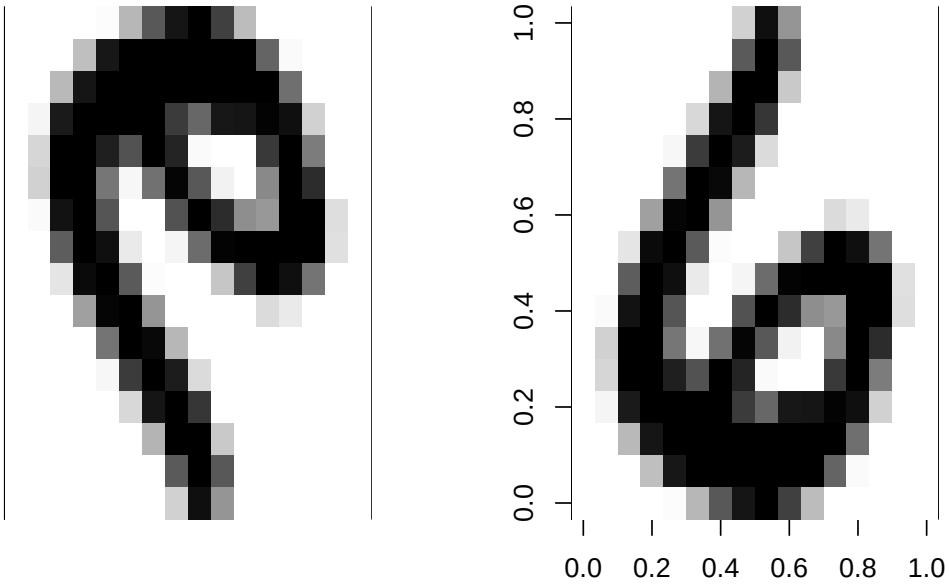



Figure 15: The left image seems to be flipped along the Y axis (left), and we've written a simple function to flip the image before plotting it (right).

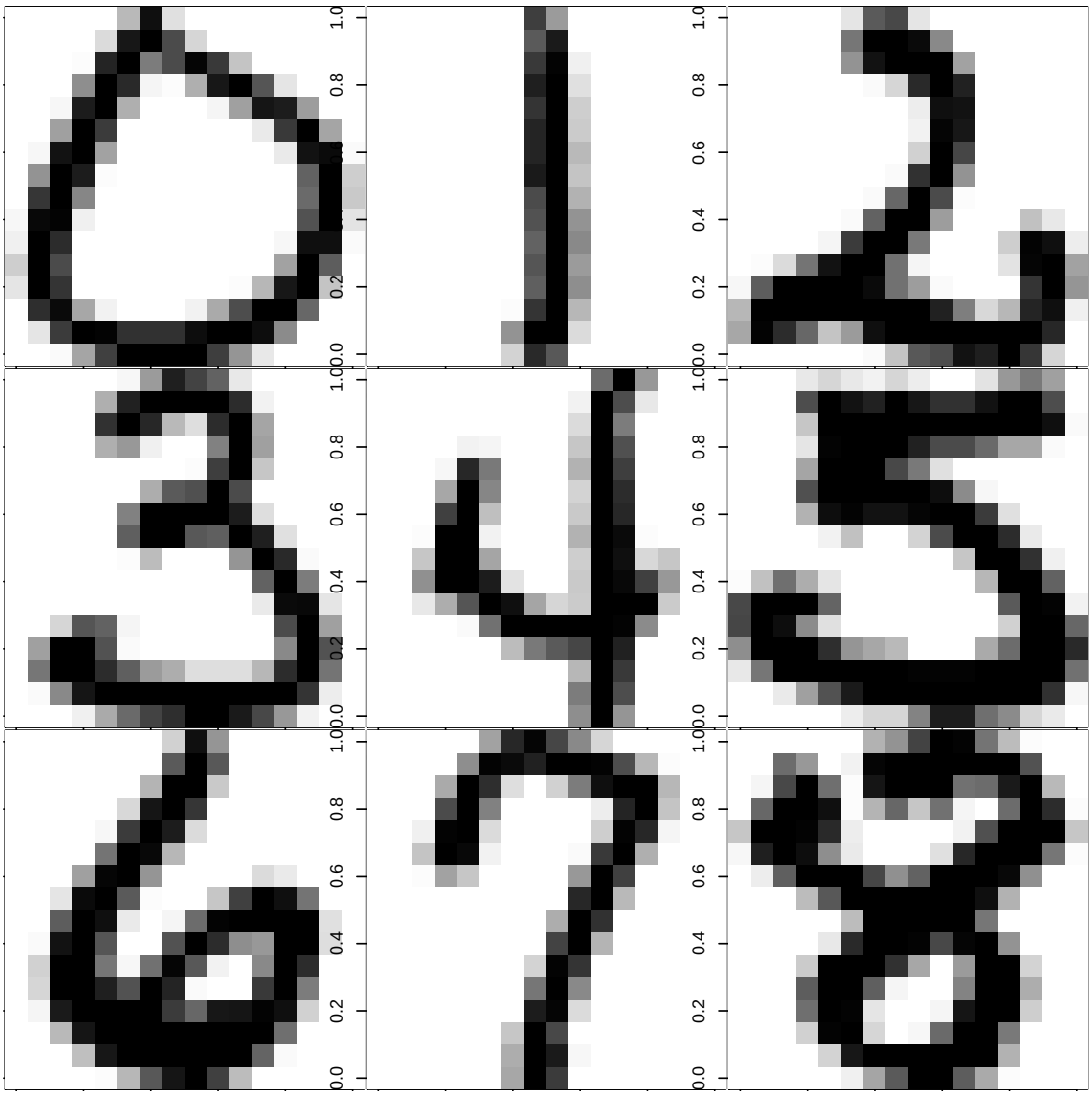
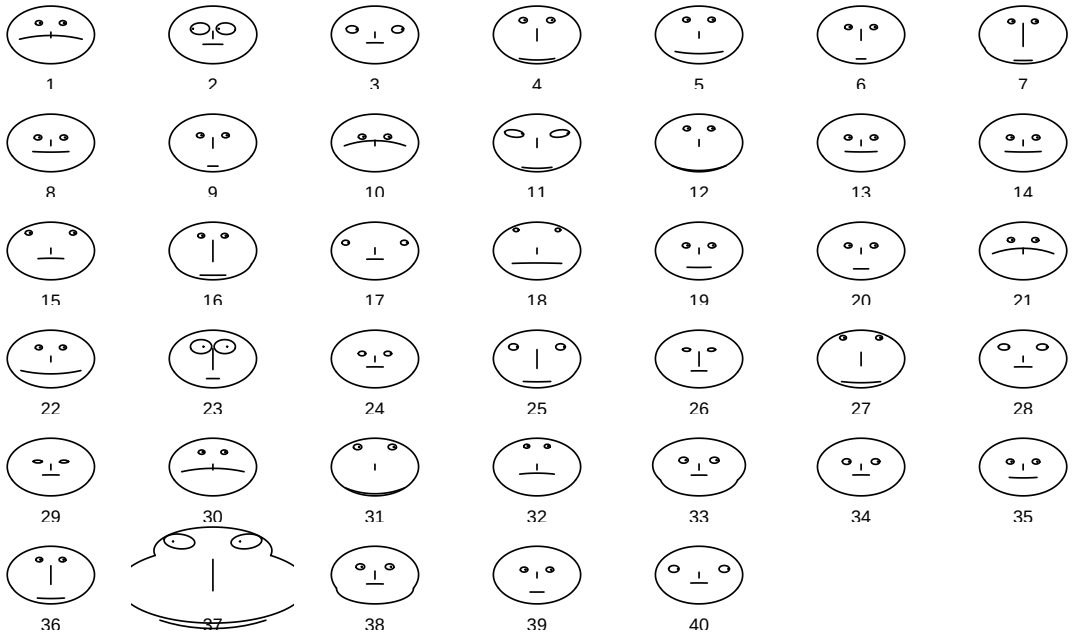
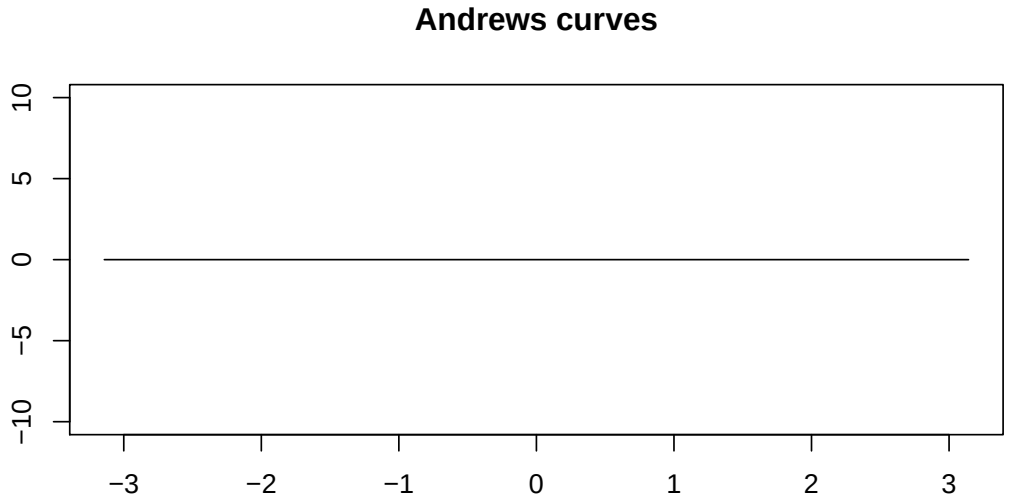


Figure 16: One example of each digit from 0 to 8.

```
## Warning in faces2(zip_code[1:40, -1]):  using only first 18 columns
of input

andrews(zip_code[1:40, -1], main='Andrews curves')
```




Only the first 18 pixels are taken into account with Chernoff faces, and Andrews curves produced nothing! We will see in the following Section on how we can visualise the `ZIP code` dataset.

# 4 Dimension Reduction Techniques

## 4.1 Multivariate moments

Suppose we have a random vector $\mathbf{X} = (X_1, \ldots, X_p)^\mathsf{T}$ arising from some probability distribution. Later in the course we shall discuss some common multivariate distributions, but here we provide some basic results regarding the means, variances and covariances.

We shall use sometimes the notation $v'$ instead of $v^\mathsf{T}$ to denote the transpose of a vector/matrix. Recall the definition of covariance for random variables $X, Y \in \mathbb{R}$ is $\mathrm{Cov}(X, Y) = \mathbb{E}\left[(X - \mathbb{E}\,X)(Y - \mathbb{E}\,Y)\right]$, which is well defined if $\mathbb{E}[X^2 + Y^2] < \infty$.

**Definition 4.1.1 (Multivariate moments).** The expected value of $\mathbf{X}$ is the $p \times 1$ vector $\mathbb{E}(\mathbf{X}) = (\mathbb{E}(X_1), \ldots, \mathbb{E}(X_p))^\mathsf{T}$, where $\mathbb{E}(X_i)$ is the usual univariate expected value of $X_i$. The **covariance matrix** of $\mathbf{X}$, written $\mathrm{Cov}\,(\mathbf{X})$, is the $p \times p$ matrix with entries $(\mathrm{Cov}\,(\mathbf{X}))_{ij} = \mathrm{Cov}\,(X_i, X_j)$. It is well-defined if $\mathbb{E}|\mathbf{X}|^2 < \infty$. If $\mathbf{Y} \in \mathbb{R}^q$, the **cross-covariance** matrix between $\mathbf{X}$ and $\mathbf{Y}$, written $\mathrm{Cov}\,(\mathbf{X}, \mathbf{Y})$, is the $p \times q$ matrix with entries $(\mathrm{Cov}\,(\mathbf{X}, \mathbf{Y}))_{ij} = \mathrm{Cov}\,(X_i, Y_j)$. It is well-defined if $\mathbb{E}\left[|\mathbf{X}|^2 + |\mathbf{Y}|^2\right] < \infty$.

One is often interested in the expected value or the covariances of linear combinations of the elements in $\mathbf{X}$.

**Proposition 4.1.2 (Moments of linear transformations).** Let $C$ be a (non-random) $q \times p$ matrix and $\mathbf{X} \in \mathbb{R}^p$ be a random vector. Then the following hold:

1. $\mathbb{E}(C\mathbf{X}) = C\,\mathbb{E}(\mathbf{X})$ (linearity)
2. $\mathrm{Cov}\,(C\mathbf{X}) = C\,\mathrm{Cov}\,(\mathbf{X})\,C^\mathsf{T}$

*Proof.* Left as an exercise. □

**Proposition 4.1.3 (and Definition of Total Variance).** Assume $\mathbf{X} \in \mathbb{R}^p$ is a random vector with $\mathbb{E}|\mathbf{X}|^2 < \infty$. The **total variance** of the random vector $\mathbf{X} \in \mathbb{R}^p$ is defined as $\mathbb{E}|\mathbf{X} - \mathbb{E}\,\mathbf{X}|^2$, and we have

$$\mathbb{E}|\mathbf{X} - \mathbb{E}\,\mathbf{X}|^2 = \sum_{i=1}^{p} \mathrm{Var}(X_i) = \mathrm{Tr}(\mathrm{Cov}(\mathbf{X})).$$

*Proof.* Left as an exercise. □

**Example 4.1.4** (Covariance of Affine Transformation)**.** Suppose $\mathbf{X} = (X_1, X_2)^\mathsf{T}$, where $X_1$ measures the average score that students obtain in the first year of college and $X_2$ that in the second year (in a scale from 0 to 10). Further suppose that $\mathbb{E}(\mathbf{X}) = (8.0, 8.2)^\mathsf{T}$ and $\mathrm{Cov}\,(\mathbf{X}) = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$. Because these two variables are

highly correlated, someone has the idea to work with the average $Y_1 = \frac{1}{2}(X_1 + X_2)$ and the increase $Y_2 = (X_2 - X_1)$. What are the mean and variance of $\mathbf{Y} = (Y_1, Y_2)$?

Clearly,

$C = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 1 \end{pmatrix}$, and $C\mathbf{X} = \begin{pmatrix} \frac{1}{2}(X_1 + X_2) \\ X_2 - X_1 \end{pmatrix}$.

Therefore,

$$\mathbb{E}(\mathbf{Y}) = C\,\mathbb{E}(\mathbf{X}) = \begin{pmatrix} \frac{1}{2}(8.0 + 8.2) \\ 8.2 - 8.0 \end{pmatrix} = \begin{pmatrix} 8.1 \\ 0.2 \end{pmatrix}$$

$$\mathrm{Cov}\,(\mathbf{Y}) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 1 \end{pmatrix} \mathrm{Cov}\,(\mathbf{X}) \begin{pmatrix} \frac{1}{2} & -1 \\ \frac{1}{2} & 1 \end{pmatrix} = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.4 \end{pmatrix}.$$

You can check that the diagonal elements in $\mathrm{Cov}\,(\mathbf{Y})$ are those you would obtain with the usual formulas for $\mathrm{Var}(X_1 + X_2) = \mathrm{Var}(X_1) + \mathrm{Var}(X_2) + 2\,\mathrm{Cov}\,(X_1, X_2)$. However, notice that $\mathrm{Cov}(\mathbf{Y})$ also tells us about the covariance between $Y_1$ and $Y_2$. In fact, it turns out that working with the average score and the increase may be a good idea, as they are uncorrelated and can hence be thought to measure complementary features (namely, overall academic performance and improvement after 1 year of studies).

---

## 4.2 Principal Component Analysis

### 4.2.1 Basic definition

Principal Component Analysis is one of the most popular descriptive tools in multivariate statistics. Suppose we are given a set of variables $\mathbf{X} = (X_1, \ldots, X_p)^\mathsf{T}$, where $p$ is too large to be able to look at all the pairwise plots between the entries of $\mathbf{X}$. One way of understanding the random vector $\mathbf{X}$ is to look at 1-dimensional summaries of it, obtained by *linear projections*, that is, to look at random variables of the form $\mathbf{v}^\mathsf{T}\mathbf{X}$, for $\mathbf{v} \in \mathbb{R}^p$. But how to choose $\mathbf{v}$? One way is to choose it such that the variance of $\mathbf{v}^\mathsf{T}\mathbf{X}$ is maximised. This is illustrated in Figure 17. But notice that $\mathrm{Var}(\alpha\mathbf{v}^\mathsf{T}\mathbf{X}) = \alpha^2\,\mathrm{Var}(\mathbf{v}^\mathsf{T}\mathbf{X})$, for $\alpha \in \mathbb{R}$. Therefore for this to make sense, we need to constrain $\mathbf{v}$ to have a fixed norm, and this is typically done by setting $|\mathbf{v}| = 1$. Once such a $\mathbf{v} \in \mathbb{R}^p, |\mathbf{v}| = 1$ maximising $\mathrm{Var}(\mathbf{v}^\mathsf{T}\mathbf{X})$ is found (let's call it $\mathbf{v}_1$), one can look for further 1-dimensional linear projections of $\mathbf{X}$, say $\mathbf{u}^\mathsf{T}\mathbf{X}$, with maximal variance, subject to the constraint that the projection is *uncorrelated* with $\mathbf{v}_1^\mathsf{T}\mathbf{X}$, and $|\mathbf{u}| = 1$.

**Definition 4.2.1 (Principal Components).** Let $\mathbf{X} \in \mathbb{R}^p$ be a random vector with $\mathbb{E}\left[\mathbf{X}^\mathsf{T}\mathbf{X}\right] < \infty$. The **1st principal component (PC) loading** is a vector $\mathbf{v}_1 \in \mathbb{R}^p, |\mathbf{v}_1| = 1$, that maximises the variance of $\mathbf{v}_1^\mathsf{T}\mathbf{X}$, or in other words

$$\mathrm{Var}(\mathbf{v}_1^\mathsf{T}\mathbf{X}) \geq \mathrm{Var}(\mathbf{u}^\mathsf{T}\mathbf{X}), \quad \forall \mathbf{u} \in \mathbb{R}^p, |\mathbf{u}| = 1.$$

For $k = 2, \ldots, p$, the **k-th principal component (PC) loading** is the

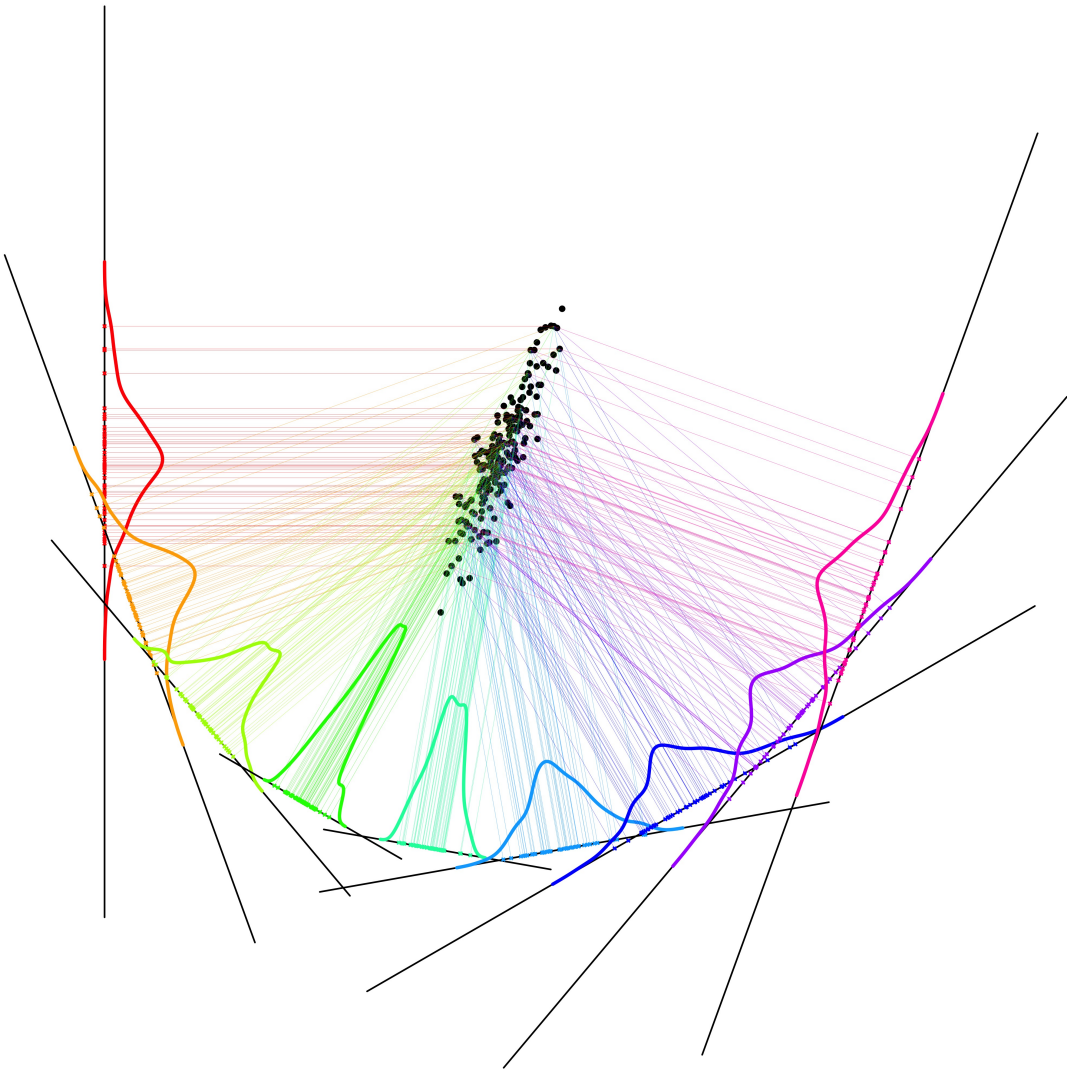


Figure 17: Several 1D projections of the data, and their respective density estimates. Notice that some of the projections are well spread, whereas some are very concentrated.

vector $\mathbf{v}_k \in \mathbb{R}^p, |\mathbf{v}_k| = 1$, that maximises $\mathrm{Var}(\mathbf{v}_k^\mathsf{T}\mathbf{X})$ subject to

$$\mathrm{Cov}(\mathbf{v}_k^\mathsf{T}\mathbf{X}, \mathbf{v}_j^\mathsf{T}\mathbf{X}) = 0,$$

for all $j = 1, \ldots, k-1$. In other words,

$$\mathrm{Var}(\mathbf{v}_k^\mathsf{T}\mathbf{X}) \geq \mathrm{Var}(\mathbf{u}^\mathsf{T}\mathbf{X}),$$

$\forall \mathbf{u} \in \mathbb{R}^p, |\mathbf{u}| = 1$ such that $\mathrm{Cov}(\mathbf{u}^\mathsf{T}\mathbf{X}, \mathbf{v}_j^\mathsf{T}\mathbf{X}) = 0$, for $j = 1, \ldots, k-1$.

The random variable $\mathbf{v}_k^\mathsf{T}\mathbf{X} \in \mathbb{R}$ is called the **k-th principal component (PC) score**.

Conducting a **principal component analysis (PCA)** means computing the PC loadings and PC scores (and doing further analysis based on these).

Some remarks are in order:

**Remark 4.2.2 (definition of PCA).**

1. Instead of putting the constraint $|\mathbf{v}| = 1$, we could have sought to maximise $\mathrm{Var}(\mathbf{v}^\mathsf{T}\mathbf{X})/|\mathbf{v}|^2$.
2. Notice that the PC loadings live in the same space as the random vector $\mathbf{X}$. The PC scores are random variables (take values in $\mathbb{R}$). You can stack them to create a vector in $\mathbb{R}^p$, but the $k$-th entry of this vector will **not** have the same meaning as the $k$-th entry of $\mathbf{X}$!
3. By construction, $\mathrm{Var}(\mathbf{v}_1^\mathsf{T}\mathbf{X}) \geq \mathrm{Var}(\mathbf{v}_2^\mathsf{T}\mathbf{X}) \geq \cdots \geq \mathrm{Var}(\mathbf{v}_p^\mathsf{T}\mathbf{X})$.

As an example, Figure 18 shows the 2 first PC loadings and corresponding scores for a dataset in 2D.

Another way of saying this is that we've transformed $\mathbf{X}$ into new variables $\mathbf{Y} = (Y_1, \ldots, Y_p)^\mathsf{T}$ that have the following properties:

- $\mathbf{Y}$ is formed by linear combinations of $\mathbf{X}$, i.e. $\mathbf{Y} = A\mathbf{X}$ for some $p \times p$ matrix $A$ whose rows have unit norm.
- $Y_k$ is uncorrelated with $Y_1, \ldots, Y_{k-1}$,
- $Y_1$ conserves as much variance in $\mathbf{X}$ as possible. Conditional on $Y_{k+1}$ being uncorrelated with $Y_1, \ldots, Y_k$, the random variable $Y_{k+1}$ conserves as much variance as possible.

### 4.2.2 Population Principal Components Analysis

In this Section we look at PCA for a random vector $\mathbf{X} \in \mathbb{R}^p$. We shall later look at PCA when one has data, that is, realizations of such random vector. Notice that $\mathrm{Var}(\mathbf{v}^\mathsf{T}\mathbf{X}) = \mathbf{v}^\mathsf{T}\,\mathrm{Cov}(\mathbf{X})\mathbf{v}$. This relates the definition of the first PC loading to the covariance matrix of $\mathbf{X}$. The next result states that the PC loadings of $\mathbf{X} \in \mathbb{R}^p$ are closely related to the spectral decomposition of the covariance matrix of $\mathbf{X}$.

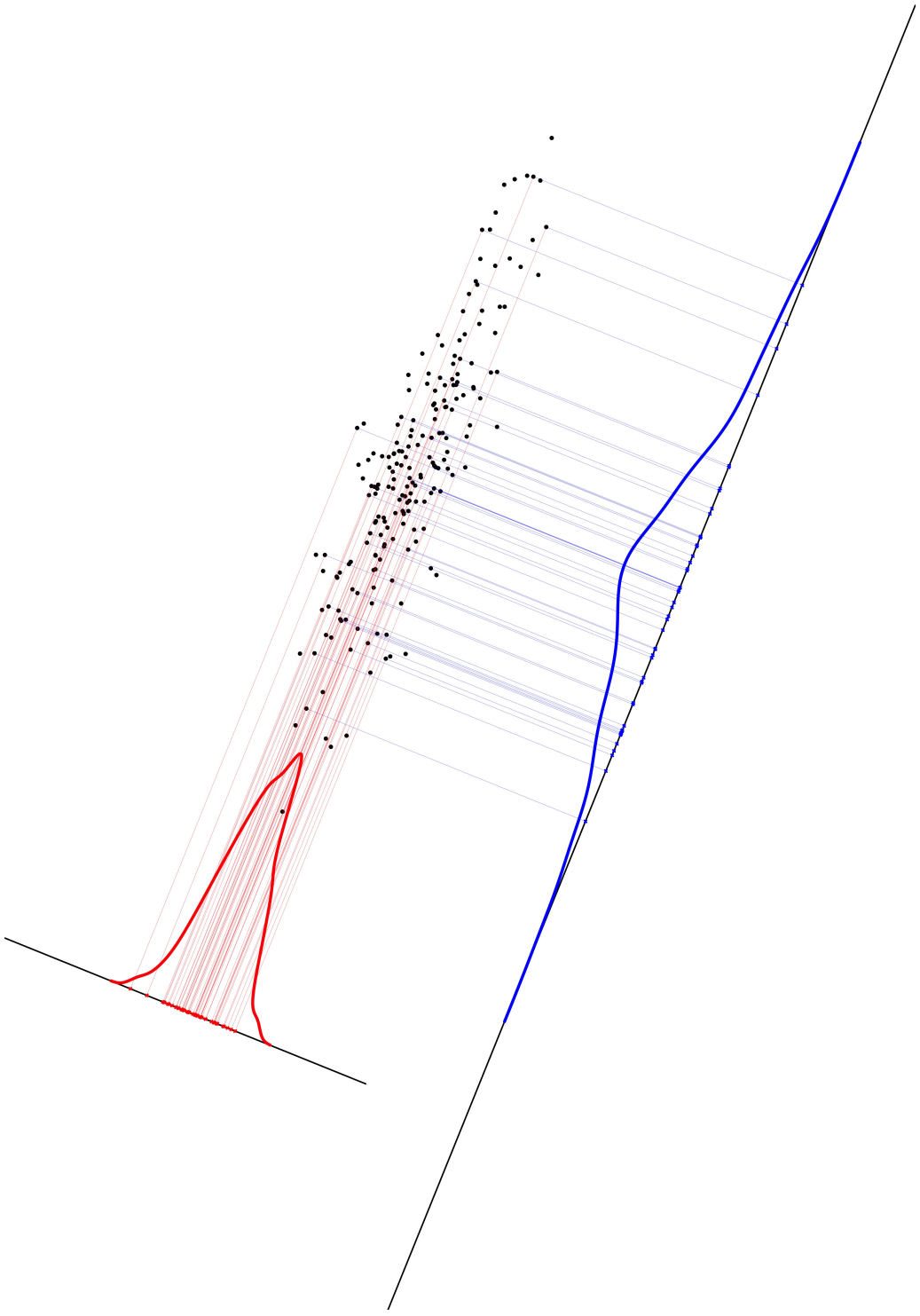


Figure 18: The 2 PC projections of the data (PC 1 in blue, PC 2 in red), and the respective density estimates of the PC scores.

**Theorem 4.2.3 (PC loadings, population version).** Let $\mathbf{X}$ be a random vector with covariance $\Sigma = \mathrm{Cov}(\mathbf{X})$, and suppose that $\Sigma$ has eigenvalues $\lambda_1 \geq \ldots \geq \lambda_p \geq 0$ and corresponding orthonormal eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_p$. Then $\mathbf{v}_1 = \mathbf{e}_1, \ldots, \mathbf{v}_p = \mathbf{e}_p$ are the PC loadings of $\mathbf{X}$, and the corresponding PC scores are $Y_1 = \mathbf{e}_1^\mathsf{T}\mathbf{X}, Y_2 = \mathbf{e}_2^\mathsf{T}\mathbf{X}, \ldots, Y_p = \mathbf{e}_p^\mathsf{T}\mathbf{X}$. Furthermore, $\mathrm{Var}(Y_1) = \lambda_1, \ldots, \mathrm{Var}(Y_p) = \lambda_p$.

*Proof.* Let us find the first principal component. The goal is to find $\mathbf{v}_1$ maximizing

$$\frac{\mathbf{v}_1^\mathsf{T}\Sigma\mathbf{v}_1}{\mathbf{v}_1^\mathsf{T}\mathbf{v}_1}.$$

Recall that the eigendecomposition allows us to reconstruct $\Sigma = E\Lambda E^\mathsf{T}$, where $E$ is a $p \times p$ orthogonal matrix ($E^\mathsf{T}E = I = EE^\mathsf{T}$) with the eigenvector $\mathbf{e}_i$ as its $i^{th}$ column. We can write the objective function as

$$\frac{\mathbf{v}_1^\mathsf{T}\Sigma\mathbf{v}_1}{\mathbf{v}_1^\mathsf{T}\mathbf{v}_1} = \frac{\mathbf{v}_1^\mathsf{T}E\Lambda E^\mathsf{T}\mathbf{v}_1}{\mathbf{v}_1^\mathsf{T}EE^\mathsf{T}\mathbf{v}_1} = \frac{\mathbf{z}_1^\mathsf{T}\Lambda\mathbf{z}_1}{\mathbf{z}_1^\mathsf{T}\mathbf{z}_1},$$

where $\mathbf{z}_1 = E^\mathsf{T}\mathbf{v}_1$ is the inner-product of $\mathbf{v}_1$ on the eigenvectors. Since $\Lambda$ is a diagonal matrix, the expression can be written as

$$\frac{\mathbf{z}_1^\mathsf{T}\Lambda\mathbf{z}_1}{\mathbf{z}_1^\mathsf{T}\mathbf{z}_1} = \frac{\sum_{i=1}^p \lambda_i z_{1i}^2}{\sum_{i=1}^p z_{1i}^2} \leq \lambda_1 \frac{\sum_{i=1}^p z_{1i}^2}{\sum_{i=1}^p z_{1i}^2} = \lambda_1.$$

That is, the maximum possible value of our objective function is $\lambda_1$, so it suffices to see that for $\mathbf{v}_1 = \mathbf{e}_1$ we attain this maximum. Because the vectors $\mathbf{e}_i$ form an orthonormal basis, for $\mathbf{v}_1 = \mathbf{e}_1$ we obtain $\mathbf{z}_1 = E^\mathsf{T}\mathbf{v}_1 = E^\mathsf{T}\mathbf{e}_1 = (1, 0, \ldots, 0)^\mathsf{T}$. Hence $\frac{\sum_{i=1}^p \lambda_i z_{1i}^2}{\sum_{i=1}^p z_{i1}^2} = \lambda_1$.

To complete the proof we proceed by induction. We want to see that $\mathbf{v}_k = \mathbf{e}_k$ maximizes $\frac{\mathbf{v}_k^\mathsf{T}E\Lambda E^\mathsf{T}\mathbf{v}_k}{\mathbf{v}_k^\mathsf{T}EE^\mathsf{T}\mathbf{v}_k}$ subject to $\mathrm{Cov}(\mathbf{v}_k^\mathsf{T}\mathbf{X}, \mathbf{v}_j^\mathsf{T}\mathbf{X}) = 0, j < k$, which is equivalent to $\mathbf{v}_k^\mathsf{T}\mathbf{v}_1 = 0$, …, $\mathbf{v}_k^\mathsf{T}\mathbf{v}_{k-1} = 0$. Because $\mathbf{v}_i = \mathbf{e}_i$ for $i < k$, the first $k-1$ components in $\mathbf{z}_k = E^\mathsf{T}\mathbf{v}_k$ are equal to zero. Therefore the objective function becomes

$$\frac{\mathbf{z}_k^\mathsf{T}\Lambda\mathbf{z}_k}{\mathbf{z}_k^\mathsf{T}\mathbf{z}_k} = \frac{\sum_{i=k}^p \lambda_i z_{ki}^2}{\sum_{i=k}^p z_{ki}^2} \leq \lambda_k,$$

hence again it suffices to see that for $\mathbf{v}_k = \mathbf{e}_k$ we attain the maximum value $\lambda_k$. We can see that this is the case, since $\mathbf{z}_k = E^\mathsf{T}\mathbf{e}_k = (0, \ldots, 1, \ldots, 0)$ (where the 1 is in the $k^{th}$ position), and hence $\frac{\sum_{i=k}^p \lambda_i z_{ki}^2}{\sum_{i=k}^p z_{ki}^2} = \lambda_k$. □

Some remarks are in order:

**Remark 4.2.4 (PCA).**

1. Notice that PC loadings (and hence PC scores) are not unique, since eigenvectors are not unique (even more so if there are repeated eigenvalues). Nevertheless we abuse of language and say "the" PC loadings.

2. If we denote $\mathbf{Y} = (Y_1, \ldots, Y_p)^\mathsf{T}$ where $Y_k = \mathbf{e}_k^\mathsf{T}\mathbf{X}$, and define the $p \times p$ matrix $E = (\mathbf{e}_1, \ldots, \mathbf{e}_p)^\mathsf{T}$, then $\mathbf{Y} = E^\mathsf{T}\mathbf{X}$, and $\mathbf{X} = E\mathbf{Y}$.

This means that there is an isometry between the PC scores and the original vector $\mathbf{X}$, and that no information is lost by working with the PC scores. In particular, $\mathbf{X}$ and $\mathbf{Y}$ have the same total variance (check this as an exercise).

3. The total variance of $\mathbf{Y}_k = (Y_1, \ldots, Y_k, 0, \ldots, 0)^\mathsf{T}$ is

$$\mathbb{E}|\mathbf{Y}_k - \mathbb{E}\,\mathbf{Y}_k|^2 = \lambda_1 + \cdots + \lambda_k,$$

and

$$\frac{\mathbb{E}|\mathbf{Y}_k - \mathbb{E}\,\mathbf{Y}_k|^2}{\mathbb{E}|\mathbf{Y} - \mathbb{E}\,\mathbf{Y}|^2} = \frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_k + \cdots + \lambda_p},$$

is the percentage of variance explained by the first $k$ PCs.

4. The PC loadings form an orthonormal basis of $\mathbb{R}^p$, and we can therefore construct (for $k = 1, \ldots, p$) the projections matrices $P_k = \mathbf{e}_1\mathbf{e}_1^\mathsf{T} + \ldots + \mathbf{e}_k\mathbf{e}_k^\mathsf{T}$, and the *rank k PC projection of* $\mathbf{X}$, $\mathbf{X}_k = P_k\mathbf{X}$. What is the total variance of $\mathbf{X}_k$? And what is the ratio of the total variance of $\mathbf{X}_k$ to the total variance of $\mathbf{X}$?

Since $Y_1, \ldots, Y_k$ represent more of the variability than $Y_{k+1}, \ldots, Y_p$, we can start by focusing our attention on the first variables. This implicitly assumes that the interesting features of the random vector $\mathbf{X}$ are contained in the directions with large variation. Obviously, in general there is going to be a loss of information due to working only with the first few principal components. Remark 4.2.4 tells us that the proportion of the total variability explained by the $k^{th}$ PC score is

$$\frac{\lambda_k}{\sum_{i=1}^p \lambda_i},$$

and the total variability jointly explained by PC scores $1, \ldots, k$ is

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i}.$$

### 4.2.3 Subspace Characterizations of Principal Component Analysis

Notice that the definition of PCA is iterative: we first find the vector that maximizes the variance of the projection of $\mathbf{X}$ along it, and then find the second vector that maximizes the variance of the projection of $\mathbf{X}$ along it, subject to the second projection being uncorrelated to the first projection, and so on. In this way we construct the random variables $Y_1, Y_2, \ldots, Y_p$ (called PC scores), and we know by Remark 4.2.4 that the total variance of $(Y_1, \ldots, Y_q)^\mathsf{T} \triangleq E_q^\mathsf{T}\mathbf{X}$, where $E_q = (\mathbf{e}_1, \ldots, \mathbf{e}_q)$, is $\lambda_1 + \cdots + \lambda_q$, the sum of the first $q$ eigenvalues of $\mathrm{Cov}(\mathbf{X})$. Since $E_q$ is a $p \times q$ matrix with orthonormal columns, $E_q^\mathsf{T}E_q = I_q$, it is natural to ask the following question: is there a $p \times q$ matrix $B$ with $B^\mathsf{T}B = I_q$ such that $B^\mathsf{T}\mathbf{X}$ has total variance larger than $\lambda_1 + \cdots + \lambda_q$? If such matrix $B$ exists, then by searching for directions $\mathbf{v} \in \mathbb{R}^p$ that iteratively maximize the variance of projections, we are doing worst than if we were to try and directly find orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_q$ such that $(\mathbf{v}_1, \ldots, \mathbf{v}_q)^\mathsf{T}\mathbf{X}$ has

maximal total variance. It turns out that fortunately, our iterative definition (of PCA) is a good one:

**Proposition 4.2.5 (Characterization of PC scores).** Assume $\mathbf{X} \in \mathbb{R}^p$ is a random vector with $\mathbb{E}|\mathbf{X}|^2 < \infty$. Let $\mathbf{e}_1, \ldots, \mathbf{e}_p \in \mathbb{R}^p$ be the (unit length) eigenvectors of $\mathrm{Cov}(\mathbf{X})$ (ordered so that their eigenvalues are decreasing), and let $E_q = (\mathbf{e}_1, \ldots, \mathbf{e}_q)$, $E_{-q} = (\mathbf{e}_{p-q+1}, \ldots, \mathbf{e}_p)$, $1 \leq q \leq p$. We have

$$\mathrm{Tr}(\mathrm{Cov}(E_{-q}^\mathsf{T}\mathbf{X})) \leq \mathrm{Tr}(\mathrm{Cov}(B^\mathsf{T}\mathbf{X})) \leq \mathrm{Tr}(\mathrm{Cov}(E_q^\mathsf{T}\mathbf{X})),$$

for all $p \times q$ matrices $B$ with $B^\mathsf{T}B = I_q$.

*Proof.* The proof uses Poincaré's inequalities (Lemma 2.4.3), see video. □

This result tells us that when looking at the coordinates of $\mathbf{X}$ onto $q$ orthonormal vectors, the total variance is maximized by taking these to be the first $q$ eigenvectors of $\mathrm{Cov}(\mathbf{X})$, but minimized when taking these to be the last $q$ eigenvectors of $\mathrm{Cov}(\mathbf{X})$.

Now suppose that instead of looking at coordinates of $\mathbf{X}$ with respect to some orthonormal vectors, we are interested in orthogonal projections of $P\mathbf{X}$ of $\mathbf{X}$. If we let $P_q = E_q E_q^\mathsf{T}$, $P_{-q} = E_{-q}E_{-q}^\mathsf{T}$, we can easily compute (do it!) that

$$\mathrm{Tr}(\mathrm{Cov}(P_q\mathbf{X})) = \lambda_1 + \cdots + \lambda_q,$$

and

$$\mathrm{Tr}(\mathrm{Cov}(P_{-q}\mathbf{X})) = \lambda_{p-q+1} + \cdots + \lambda_p.$$

Notice that $P_q, P_{-q}$ are both orthogonal projections with trace $q$ (or rank $q$). The following result tells us that in terms of total variance (or percentage of variance), the best (respectively worst) orthogonal projection with trace $q$ is given by $P = P_q$ (respectively $P = P_{-q}$).

**Proposition 4.2.6 (Characterization of PCA by Orthogonal Projections).** Under the assumptions and notation of Proposition 4.2.5, if $P$ is a $p \times p$ orthogonal projection matrix with $\mathrm{Tr}(P) = q$, then

$$\mathrm{Tr}(\mathrm{Cov}(P_{-q}\mathbf{X})) \leq \mathrm{Tr}(\mathrm{Cov}(P\mathbf{X})) \leq \mathrm{Tr}(\mathrm{Cov}(P_q\mathbf{X}))$$

that is, the total variance of $P\mathbf{X}$ is less than or equal to the total variance of $P_q\mathbf{X}$, but greater than or equal the total variance of $P_{-q}\mathbf{X}$.

*Proof.* The proof is based on Proposition 4.2.5, see video. □

Now suppose that instead of being concerned with maximizing the total variance of the projection, we are trying to find the best rank $q$ orthogonal projection $P$ such that $P\mathbf{X}$ is closest to $\mathbf{X}$. This can be thought of as the best $q$ dimensional approximation of $\mathbf{X}$, since $P\mathbf{X}$ lives on the $q$-dimensional image of $P$. Again, it turns out that the best and worst projections (in this sense) are given by the projections $P = P_q, P = P_{-q}$, obtained from the eigenvectors of $\mathrm{Cov}(\mathbf{X})$.

**Proposition 4.2.7 (Characterization of PCA by Approximations).** Under the assumptions and notation of Proposition 4.2.5, and $\mathbb{E}\,\mathbf{X} = 0$, if $P$ is a $p \times p$ orthogonal projection matrix with $\mathrm{Tr}(P) = q$, then

$$\mathbb{E}|\mathbf{X} - P_{-q}\mathbf{X}|^2 \geq \mathbb{E}|\mathbf{X} - P\mathbf{X}|^2 \geq \mathbb{E}|\mathbf{X} - P_q\mathbf{X}|^2.$$

*Proof.* The proof is based on Proposition 4.2.5, see exercise. □

### 4.2.4 Sample Principal Components

Recall that the sample variance of $y_1, \ldots, y_n \in \mathbb{R}$ is given by $\sum_{i=1}^n (y_i - \overline{y})^2/(n-1)$, and the sample covariance between $y_1, \ldots, y_n \in \mathbb{R}$ and $z_1, \ldots, z_n \in \mathbb{R}$, where $y_i, z_i$ are paired, is given by $\sum_{i=1}^n (y_i - \overline{y})(z_i - \overline{z})/(n-1)$, where $\overline{y} = \sum_i y_i/n$ and similarly for $\overline{z}$.

So far we focused on the population case, or random vector case, where we assume we know the covariance matrix $\Sigma$. In practice, we get data $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{iid}}{\sim} \mathbf{X} \in \mathbb{R}^p$, and these are stacked in a *data matrix* $\mathtt{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\mathsf{T}$. The first step for computing that sample covariance matrix is to *center the columns* of $\mathtt{X}$. If $H = I_n - \mathbf{1}\mathbf{1}^\mathsf{T}/n$, $H\mathtt{X}$ is a column-centered version of $\mathtt{X}$ (see exercises). Define $S = \mathtt{X}^\mathsf{T} H\mathtt{X}/(n-1)$, the sample covariance of $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

**Definition 4.2.8 (Sample PC loadings).** The first sample PC loading is the unit vector $\mathbf{v}_1 \in \mathbb{R}^p$ that maximises the sample variance of $\mathbf{v}_1^\mathsf{T}\mathbf{x}_1, \ldots, \mathbf{v}_1^\mathsf{T}\mathbf{x}_n$. The $k$-th sample PC loading is the unit vector $\mathbf{v}_k \in \mathbb{R}^p$ that maximises the sample variance of $\mathbf{v}_k^\mathsf{T}\mathbf{x}_1, \ldots, \mathbf{v}_k^\mathsf{T}\mathbf{x}_n$ subject to $(\mathbf{v}_k^\mathsf{T}\mathbf{x}_i)_{i=1,\ldots,n}$ and $(\mathbf{v}_j^\mathsf{T}\mathbf{x}_i)_{i=1,\ldots,n}$ being uncorrelated for $j = 1, \ldots, k-1$.

Is is an straightforward exercise (do it!) to show that the covariance between $(\mathbf{v}_k^\mathsf{T}\mathbf{x}_i)_{i=1,\ldots,n}$ and $(\mathbf{v}_j^\mathsf{T}\mathbf{x}_i)_{i=1,\ldots,n}$ is given by $\mathbf{v}_k^\mathsf{T} S\mathbf{v}_j$. Therefore the definition of the sample PC loadings can be given as follows: The first sample PC loading is the vector that maximises $\mathbf{v}^\mathsf{T} S\mathbf{v}$. A $k$-th sample PC loading is the unit vector $\mathbf{v}_k \in \mathbb{R}^p$ that maximises $\mathbf{v}_k^\mathsf{T} S\mathbf{v}_k$ subject to $\mathbf{v}_k^\mathsf{T} S\mathbf{v}_j = 0$ for $j = 1, \ldots, k-1$. Mimicking the proof of Theorem 4.2.3, we get the following result.

**Proposition 4.2.9 (Sample PC loadings).** The eigenvectors $\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_p$ of the sample covariance matrix $S$ are the (sample) PC loadings of the data $\mathtt{X}$. The observations $(\hat{\mathbf{e}}_k^\mathsf{T}\mathbf{x}_j)_{j=1,\ldots,n}$ are the $k$-th (sample) PC scores, and their variance is $\hat{\lambda}_k$, the eigenvalue of $S$ corresponding to $\hat{\mathbf{e}}_k$. The proportion of variance explained by PC $k$ is $\hat{\lambda}_k / \sum_{j=1}^p \hat{\lambda}_j$.

*Proof.* Left as an exercise. □

In practice, we drop the word "sample" and we just talk about PC loadings and PC scores. Also, people usually refer to the PC scores as the "principal components" (**not** "principle components"). It is however important to recall the PC scores alone don't give the complete picture of the data, and that the PC loadings are needed

to reconstruct the data from the PC scores. Letting $E$ be the matrix with columns equal to orthonormal eigenvectors of $S$, we notice that the matrix $\mathtt{Y} = \mathtt{X}E$ has as $k$-th column the $k$-th PC scores. Furthermore, since $EE^\mathsf{T} = I_p$, we have $\mathtt{X} = \mathtt{Y}E^\mathsf{T}$. Taking row $i$ of this expression, we get

$$\mathbf{x}_i = \sum_{k=1}^{p} (\mathtt{Y})_{ik} \hat{\mathbf{e}}_k, \tag{4.2.1}$$

that is, *we can express each observation as a linear combination of the PC loadings, with weights given by the PC scores!*

Notice that since the sample covariance matrix $S$ is only an estimator of the true covariance matrix $\Sigma$, the eigenpairs $(\hat{\mathbf{e}}_i, \hat{\lambda}_i)$ are only estimators of the true eigenpairs $(\mathbf{e}_i, \lambda_i)$. We will discuss the sampling distribution of the eigenstructure of the sample covariance in Section 4.2.8. See also the video which shows that $S$ is an unbiased estimator of $\Sigma$.

Importantly, there is a link between the PC scores and the spectral decomposition of $\mathtt{X}\mathtt{X}^\mathsf{T}$, and a link between PC loadings and scores and the SVD of $\mathtt{X}$, see exercises.

### 4.2.5 Deciding the number of principal components

Recall that $\mathtt{X} = \mathtt{X}EE^\mathsf{T} = \mathtt{Y}E^\mathsf{T}$, as seen above. Let $E_q$ be the $p \times q$ matrix with columns the first $q$ columns of $E$. Then $\mathtt{X}_q \triangleq \mathtt{X}E_qE_q^\mathsf{T}$ is a rank $q$ approximation to $\mathtt{X}$, and we can also see that $\mathtt{X}_q = \mathtt{Y}_qE^\mathsf{T}$, where $\mathtt{Y}_q$ is the same matrix as $\mathtt{Y}$, but with the last $p-q$ columns replaced by the zero vector. Notice that the $i$-th row of $\mathtt{X}_q$ is given by

$$\mathbf{x}_i = \sum_{k=1}^{q} (\mathtt{Y})_{ik} \hat{\mathbf{e}}_k,$$

which is a truncation of (4.2.1). This tells us that we can approximate the data using $q$ PCs, and the approximation error is (do the calculations!) given by

$$\|\mathtt{X} - \mathtt{X}_q\|_F^2 = (n-1) \cdot (\hat{\lambda}_{q+1} + \cdots + \hat{\lambda}_p).$$

The crucial question is "**how do I choose $q$?**". The larger the $q$, the better approximation you get, but the lower the $q$, the more parsimonious the representation (and hence the easier it is to interpret the resulting PC scores and loading, since there are fewer of them). We now give two commonly used rules for choosing the number of PCs:

**The 90% rule-of-thumb** Recall that the variance of the $i^{th}$ principal component is $\hat{\lambda}_i$ and the proportion of explained variance is $\hat{\lambda}_i / \sum_{i=1}^{p} \hat{\lambda}_i$. A possible rule-of-thumb is to choose the smallest number $k$ of components that explain 90% of the variance, that is

$$\frac{\sum_{i=1}^{k} \hat{\lambda}_i}{\sum_{i=1}^{p} \hat{\lambda}_i} > 0.9.$$

Of course, 0.9 is an arbitrary choice and can be changed depending on the application.

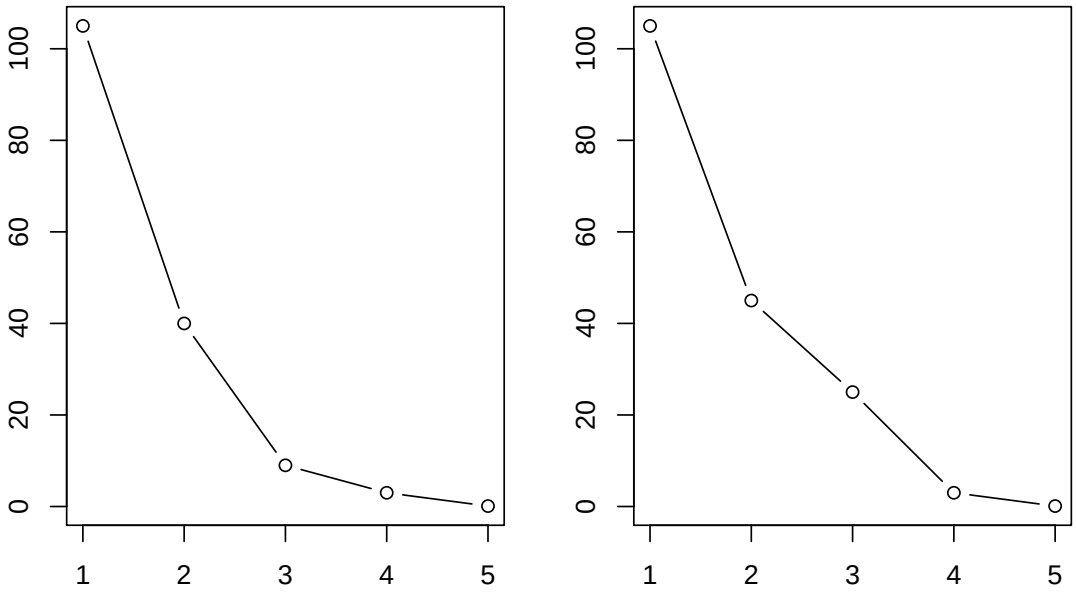



Figure 19: 2 scree plots. Remember that the "y" axis represents the variance, and not the standard deviation.

**The shoulder rule** A graphical alternative to the rule-of-thumb is to produce the so-called **scree plot**, which displays $(i, \hat{\lambda}_i)$ in a plot, that is, a plot of the PC score variances (and **not the standard deviations!**). The left panel in Figure 19 shows a scree plot for $\hat{\lambda} = (105, 40, 9, 3, 0.1)^\intercal$. The idea is to look for a "shoulder" in the plot, that is a value where the remaining $\hat{\lambda}_i$s flatten out. Here we see that the plot flattens out for $\hat{\lambda}_3, \hat{\lambda}_4, \hat{\lambda}_5$, which would suggest keeping only the first two PCs (you keep PCs **before** the shoulder). For two PCs the proportion of explained variance is (105+40)/(105+40+9+3+0.1)=0.923. The right panel in Figure 19 shows the scree plot for $\hat{\lambda} = (105, 45, 25, 3, 0.1)^\intercal$. Here the plot flattens out after $\hat{\lambda}_4$, which suggests keeping three PCs The proportion of explained variance with 3 PCs is 0.983, and with 2 PCs 0.842.

These rules are quite subjective. Interesting features of the data can be in the first few PCs, which are kept (and hence analysed) by one of the two rules given above, but it may also be that interesting features of the data are hidden in PCs that are not kept by either rule.

### 4.2.6 Interpreting the principal components

Recall that we observe $\mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{iid}}{\sim} \mathbf{X} = (X_1, \ldots, X_p)^\intercal \in \mathbb{R}^p$. While the original variables $X_1, \ldots, X_p$ usually have some context-specific meaning, principal components are linear combinations of these that require some effort to interpret. Let us start by seeing a couple of examples.

**Example 4.2.10** (Computation and interpretation of PC scores)**.** Let us go back to the student grades of Example 4.1.4, where the covariance for first and second

year grades was

$$\mathrm{Cov}\,(\mathbf{X}) = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}.$$

Its eigenvectors are $\mathbf{e}_1 = (1/\sqrt{2}, 1/\sqrt{2})^\mathsf{T}$ and $\mathbf{e}_2 = (-1/\sqrt{2}, 1/\sqrt{2})^\mathsf{T}$, with corresponding eigenvalues $\lambda_1 = 1.8$, $\lambda_2 = 0.2$. The principal components are therefore $\mathbf{e}_1^\mathsf{T}\mathbf{X} = (X_1 + X_2)/\sqrt{2}$ and $\mathbf{e}_2^\mathsf{T}\mathbf{X} = (X_2 - X_1)/\sqrt{2}$. That is, the first PC (score) is proportional to the average grade and the second PC (score) to the difference between grades. The proportion of variability explained by the first component is $1.8/(1.8 + 0.2) = 0.9$.

Now suppose that the first year grades are more variable than the second year grades, so that

$$\mathrm{Cov}\,(\mathbf{X}) = \begin{pmatrix} 4 & 1.6 \\ 1.6 & 1 \end{pmatrix}.$$

Notice that the correlation between $(X_1, X_2)$ is still $1.6/(\sqrt{4}\sqrt{1}) = 0.8$ as before. The eigenvectors now are $\mathbf{e}_1^\mathsf{T} = (0.917, 0.397)$ and $\mathbf{e}_2^\mathsf{T} = (-0.397, 0.917)$ with corresponding eigenvalues $\lambda_1 = 4.69$, $\lambda_2 = 0.307$. Hence the principal components are

$$\mathbf{e}_1^\mathsf{T}\mathbf{X} = 0.917X_1 + 0.397X_2,$$

and

$$\mathbf{e}_2^\mathsf{T}\mathbf{X} = 0.917X_2 - 0.397X_1.$$

As before the first principal component is found by adding up $X_1$ and $X_2$, but now $X_1$ is assigned a higher weight than $X_2$ because $X_1$ explains more variability than $X_2$ (since $\mathrm{Var}(X_1) > \mathrm{Var}(X_2)$). Similarly, the second principal component is a weighted difference of $X_2$ minus $X_1$. The proportion of variability explained by the first component is $4.69/(4.69 + 0.307) = 0.938$, which is higher than before.

---

**Example 4.2.11** (Computation and interpretation of PC scores)**.** Consider the following covariance matrix of 5 variables

$$\Sigma = \begin{pmatrix} 1 & .9 & .5 & .1 & .1 \\ .9 & 1 & .5 & .1 & .1 \\ .5 & .5 & 1 & .5 & .5 \\ .1 & .1 & .5 & 1 & .9 \\ .1 & .1 & .5 & .9 & 1 \end{pmatrix}.$$

Let us find eigenvectors and eigenvalues using R.

```
## define and print Sigma
(Sigma <- matrix(c(1,.9,.5,.1,.1, .9,1,.5,.1,.1, .5,.5,1,.5,.5,
                   .1,.1,.5,1,.9,
                .1,.1,.5,.9,1), nrow=5, byrow=TRUE))

##      [,1] [,2] [,3] [,4] [,5]
## [1,]  1.0  0.9  0.5  0.1  0.1
## [2,]  0.9  1.0  0.5  0.1  0.1
```

```
## [3,]  0.5  0.5  1.0  0.5  0.5
## [4,]  0.1  0.1  0.5  1.0  0.9
## [5,]  0.1  0.1  0.5  0.9  1.0

## define and print rounded lambda (the variances)
op <- options(digits = 2) # print only 2 digits
(lambda <- eigen(Sigma)$values)

## [1] 2.69 1.70 0.41 0.10 0.10

## define and print rounded version of E, the PC loadings
E <- eigen(Sigma)$vectors
rownames(E) = paste('V',1:5, sep='')
colnames(E) <- paste0('PC', 1:5)
E

##      PC1      PC2   PC3      PC4      PC5
## V1 -0.43  5.0e-01 -0.25  0.0e+00  7.1e-01
## V2 -0.43  5.0e-01 -0.25  3.9e-17 -7.1e-01
## V3 -0.51 -1.1e-15  0.86 -7.4e-17  2.2e-16
## V4 -0.43 -5.0e-01 -0.25 -7.1e-01  1.7e-16
## V5 -0.43 -5.0e-01 -0.25  7.1e-01  1.4e-16

## Percentage of variance explained by first 2 PCs
( (lambda[1]+lambda[2])/sum(lambda))

## [1] 0.88

options(op); # restore print settings
```

Here all entries in the first eigenvector have the same sign (first column in `E`), and in fact have a similar value. The first PC is computed as $Y_1 = -(0.43X_1 + 0.43X_2 + 0.51X_3 + 0.43X_4 + 0.43X_5)$, which roughly speaking is proportional to the mean across $X_1, \ldots, X_5$. Hence the first PC is approximately an average of the variables in $\mathbf{X}$.

The second PC is $Y_2 = 0.5(X_1 + X_2) - 0.5(X_4 + X_5)$, which is the difference between the mean of each block of variables.

---

**Example 4.2.12** (cars dataset)**.** Consider the cars dataset, which measures 7 continuous characteristics for 32 cars. Below we load the data and compute the PC loadings.

```
require(grDevices)
op <- options(digits = 2) # print only 2 digits
data(mtcars)
nn <- rownames(mtcars)
```

```
cars_pca <- prcomp(mtcars[,1:7])
cars_pca$rot
```

```
##           PC1     PC2    PC3     PC4     PC5     PC6     PC7
## mpg  -0.0381  0.0092  0.990  0.0678 -0.0910  0.0475  0.0534
## cyl   0.0120 -0.0034 -0.063 -0.2701 -0.9155  0.1954 -0.2157
## disp  0.8996  0.4355  0.032 -0.0015  0.0085 -0.0035 -0.0031
## hp    0.4348 -0.8998  0.023  0.0283 -0.0015 -0.0021  0.0006
## drat -0.0027 -0.0039  0.040 -0.0377  0.3099  0.6154 -0.7226
## wt    0.0062  0.0049 -0.084  0.1701 -0.0286  0.7598  0.6212
## qsec -0.0067  0.0250 -0.073  0.9441 -0.2379 -0.0598 -0.2064
```

```
options(op)
```

We can notice that the first PC score is approximately a weighted average of `disp` and `hp` variables, with the other variables having little to no contribution. The second PC score is a contrast between `disp` and `hp`, with the other variables having little to no contribution. The third PC is essentially `mpg`. The 4th PC is essentially a contrast between a weighted average of `qsec, wt, mpg` and `cyl`.

---

From these examples we get a couple of important observations:

**Remark 4.2.13 (Interpretation of PC scores).**

1. The results of the PCA can change substantially when the variance of an input variable $X_i$ changes. We will discuss this issue in the Section 4.2.7.

2. In order to interpret the meaning of the PC scores, it is important to look at the entries of the PC loadings (the eigenvectors of $\mathrm{Cov}(X)$). The entries tell us about the contribution of each variable $X_i$ to any given PC score $Y_k$. If $\mathbf{e}_k = (e_{k1}, \ldots, e_{kp})^\mathsf{T}$, then PC score $k$ is

$$Y_k \triangleq \mathbf{e}_k^\mathsf{T}\mathbf{X} = \sum_{i=1}^{p} e_{ki}X_i.$$

   Large positive $e_{ki}$ indicates that $X_i$ has a positive contribution to $Y_k$, large negative $e_{ki}$ that the contribution is negative, and $e_{ki}$ close to 0 means that variable $X_i$ contributes little to that component. This is an *exact interpretation* of PC scores. Another way to interpret PC scores is to look at the expression

$$\mathbf{X} = \sum_{k=1}^{p} Y_k\mathbf{e}_k.$$

Larger $Y_k$ means larger contribution of $\mathbf{e}_k$ in $\mathbf{X}$, so if $\mathbf{e}_1 = (0.8, -0.6)^\mathsf{T}$ (say) where $\mathbf{X} = (\text{“age”}, \text{“height”})^\mathsf{T}$, then an observation with larger $Y_1$ (PC scores 1) could be interpreted as having older age but smaller height. We call such interpretation an *approximate interpretation* since the previous statement is only correct if all other PC scores are kept fixed.

3. The magnitude of $e_{ki}$ does **not** correspond to the value of the correlation between the PC $k$ and variable $X_i$, see Proposition 4.2.14 below. Therefore, when interpreting PC scores, be cautious in the terminology you use: you can use terms like “weighted average”, “contrast”, “essentially”, as exemplified in the examples above.

4. In some sense PC scores are weird, because they are linear combinations of the original variables, which could be totally incomparable: for instance $X_1$ could be the miles/gallon consumption of a car, and $X_2$ could be the number of cylinders of the car. Then what is the unit of a linear combination of $X_1, X_2$? If one is in such a situation, then it is best to perform PCA on standardized variables, as explained in Section 4.2.7. However, if all variables are in the same unit, then taking a linear combination of them makes more sense, and you shouldn't standardize variables.

As mentioned in the remark, the value of the $e_{ki}$ is not equal to the correlation between PC score $k$ and variable $X_i$, but is related according to the following formula.

**Proposition 4.2.14 (Correlation between variables and PC scores).** The correlation between $X_i$ and $Y_k$ is equal to

$$\rho_{X_i,Y_k} = \frac{e_{ki}\sqrt{\lambda_k}}{\sqrt{\sigma_{ii}}}.$$

*Proof.* Left as an exercise. □

Hence, the contribution of variable $i$ on component $k$ can either be measured by $e_{ki}$ or $\rho_{X_i,Y_k}$. The latter is less preferable, as it only measures univariate correlation. Instead $e_{k1}, \ldots, e_{kp}$ can be interpreted as coefficients in a multiple regression, where coefficients tell us the importance of one variable in the presence of all the remaining variables.

### 4.2.7 Principal component analysis on standardized variables

As we illustrated in Example 4.2.10, principal components are affected by changes in $\text{Var}(X_i)$. For instance, the PCA results are affected when we change the units of measurement for $X_i$: measuring a distance in meters rather than kilometers means we multiply the variable by 1000, and hence increase the variance by a factor $10^6$.

Thus, $X_i$ measured in meters will have a much greater effect on the principal components than when measured in kilometers! Furthermore, the interpretation of PC scores is a bit puzzling when variables are not measured in the same unit (or represent different types of quantities), as mentioned in Remark 4.2.13.

Whether this is an issue or not depends on the specific application. As a general rule, we want to avoid the results being sensitive to the scale in the input measurements, but there are some cases in which we may want to consider that variables with higher variance are more informative. For instance, this is the case when $X_i$ is observed in two different groups, so that the total $\mathrm{Var}(X_i)$ is the sum of the between groups variance plus the within-groups variance. In this situation, $\mathrm{Var}(X_i)$ tends to be larger whenever there are differences between groups, and it may be desirable that these variables have a higher weight on the principal components. Another example is when all $X_i$s are measured on the same units of measurement, e.g. when measuring blood pressure at several points over time.

In any case, whenever we want to give the same weight (a priori) to all variables, an easy fix is to work with standardized variables $Z_i = X_i/\sqrt{\sigma_{ii}}$, which ensures that $\mathrm{Var}(Z_1) = \cdots = \mathrm{Var}(Z_p) = 1$. In matrix notation, $\mathbf{Z} = (Z_1, \ldots, Z_p)^\intercal = V^{-1/2}\mathbf{X}$ where $V = \mathrm{diag}(\mathrm{Var}(X_1), \ldots, \mathrm{Var}(X_p))$, the diagonal matrix with entries the variances of $X_i$s. It is easy to see that working with standardized variables is equivalent to obtaining principal components on the correlation matrix (rather than the covariance matrix).

$$\mathrm{Cov}\,(\mathbf{Z}) = \mathrm{Cov}\left(V^{-1/2}\mathbf{X}\right) = V^{-1/2}\,\mathrm{Cov}\,(\mathbf{X})\,V^{-1/2} = \mathrm{Cor}(\mathbf{X}),$$

where $\mathrm{Cor}(\mathbf{X})$ is the correlation matrix of $\mathbf{X}$. The total variability in $\mathbf{Z}$ is $\mathrm{Tr}(\mathrm{Cov}\,(\mathbf{Z})) = \sum_{i=1}^{p} 1 = p$, hence the proportion of explained variability by the first $k$ components is

$$\frac{\sum_{i=1}^{k} \lambda_i}{p},$$

where $\lambda_i$ is the $i$-th largest eigenvalue of the correlation matrix of $\mathbf{X}$. In the sample world, the data matrix with standardized variables is $\mathtt{Z} = \mathtt{X}V^{-1/2}$, where $V$ is the diagonal matrix with the sample variances in the diagonal, that is the $i$-th diagonal entry of $V$ is $\left(\mathtt{X}^\intercal H \mathtt{X}/(n-1)\right)_{ii}$. We now give an example to illustrate how standardizing the variables changes the PC loadings (and hence scores).

**Example 4.2.15** (Link between standardized and non-standardized PCs)**.** Suppose we observe the random sample from $\mathbf{X} = (X_1, X_2)^\intercal$ shown in Figure 20 (top left). The two variables are highly positively correlated, in fact the sample correlation coefficient is 0.88, the sample variance of $X_1$ is 0.86 and the sample variance of $X_2$ is 4.39. The covariance is $0.88 \cdot \sqrt{0.86} \cdot \sqrt{4.39} = 1.71$ The eigenvectors (PC loadings) $\mathbf{v}_1$ and $\mathbf{v}_2$ are shown in solid red and dashed red, respectively. On the top right subfigure of Figure 20, we see the same data after having standardized the variables to have variance 1. We now see that the PC loadings are different to those in the top left subfigure. In the bottom subfigures of Figure 20, we see the PC scores for original (non-standardized) variables (bottom left) and standardized variables (bottom right). Notice that we cannot go from one to the other just by scaling the axes. The code for generating the figure is given below.

```r
plot_vector <- function(v, alpha=1, ...){
    lines(alpha*matrix(c(-v[1], v[1], -v[2], v[2]), 2,2), ...)
}

set.seed(1)
x = rnorm(200)
y = rnorm(200) + 2*x
X = cbind(x,y)

op <- par(mfcol=c(2,2))
par(mai=c(.4,.7,.3,.1))
plot(X, cex=.6, asp=1, xlim=c(-2,2), ylim=c(-5, 5), pch=1, axes=TRUE,
     main='Non-standardized variables')
abline(h=0, v=0, lty=3)
Xpca <- prcomp(X)
plot_vector(Xpca$rot[,1], alpha=10, lwd=2, col=2)
plot_vector(Xpca$rot[,2], alpha=10, lty=2, lwd=2, col=2  )
par(mai=c(.7,.7,.1,.1))
plot(Xpca$x, cex=.6, asp=1, xlim=c(-2,2), ylim=c(-5, 5), pch=1, axes=TRUE)
abline(h=0, v=0, lty=3)
par(mai=c(.4,.7,.3,.1))
plot(scale(X), cex=.6, asp=1, xlim=c(-2,2), ylim=c(-5, 5), pch=1, axes=TRUE,
     yaxt='s', main='Standardized variables')
abline(h=0, v=0, lty=3)
Xpca <- prcomp(X, scale=TRUE)
plot_vector(Xpca$rot[,1], alpha=10, lwd=2, col=2)
plot_vector(Xpca$rot[,2], alpha=10, lty=2, lwd=2, col=2  )
par(mai=c(.7,.7,.1,.1))
plot(Xpca$x, cex=.6, asp=1, xlim=c(-2,2), ylim=c(-5, 5), pch=1, axes=TRUE)
abline(h=0, v=0, lty=3)
par(op)
```

### 4.2.8 Sampling Properties of Principal Components (not examinable)

Deriving the sampling distribution for $(\hat{\lambda}_i, \hat{\mathbf{e}}_i)$ is complex and requires numerous assumptions on the distribution of $\mathbf{X}$ and the true values of $\lambda_1, \ldots, \lambda_p$. Theorem 4.2.16 considers the perhaps most commonly made assumptions of underlying normality and eigenvalue multiplicity being equal to 1.

**Theorem 4.2.16 (Asymptotic distribution of eigendecomposition).** Assume that $\lambda_1 > \ldots > \lambda_p > 0$ and $\mathbf{x}_i$ are independent observations from a multivariate Normal distribution. Let $\hat{\lambda_k}$ and $\hat{\mathbf{e}}_k$ be the $k^{th}$ eigenvector

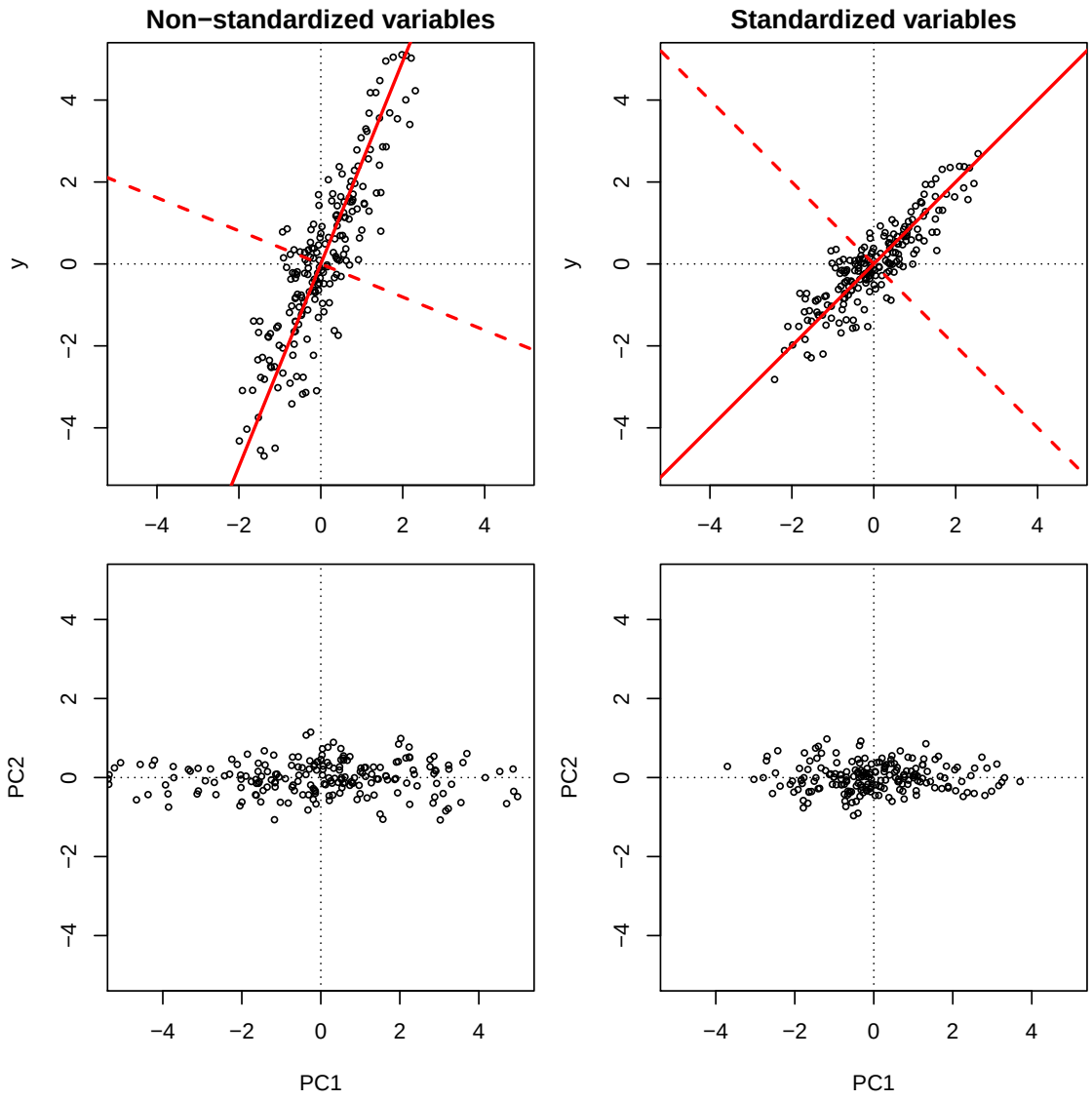



Figure 20: Dataset with non-standardized variables (top left) and standardized variables (right). The thick solid red line represents PC loading 1, and the thick dashed red line represents PC loading 2. Notice that they are different in the two plots. The bottom figures represent the PC scores 1 vs. 2 for non-standardized variables (bottom left) and standardized variables (bottom right).

and eigenvalue of the sample covariance matrix. Choosing $\hat{\mathbf{e}}_k$ such that $\hat{\mathbf{e}}_k^\mathsf{T}\mathbf{e}_k \geq 0$, we have

1. $\sqrt{n}(\hat{\lambda}_k - \lambda_k) \xrightarrow{d} N(0, 2\lambda_k^2)$. Further $\hat{\lambda}_1, \cdots, \hat{\lambda}_p$ are independent.
2. $\sqrt{n}(\hat{\mathbf{e}}_k - \mathbf{e}_k) \xrightarrow{d} N_p\left(0, \sum_{j\neq k} \frac{\lambda_j\lambda_k}{(\lambda_j-\lambda_k)^2}\mathbf{e}_j\mathbf{e}_j^\mathsf{T}\right)$
3. Each $\hat{\lambda}_k$ is independent of the corresponding $\hat{\mathbf{e}}_k$.

It is important to emphasize that Theorem 4.2.16 gives the asymptotic distribution of the sample eigenstructure, that is when $n \to \infty$. For instance, we know that $\hat{\lambda}_i > 0$ (since the sample covariance matrix is positive definite), whereas its asymptotic Normal distribution assigns non-zero probability to $\hat{\lambda}_i < 0$, hence for small and even moderately large $n$ the accuracy of the asymptotic approximation is often suspect. Nevertheless, when the assumptions of the Theorem are reasonable, we can obtain approximate $1-\alpha$ confidence intervals as $\hat{\lambda}_i \pm z_{\alpha/2}\sqrt{2\hat{\lambda}_i^2/n}$, where $z_\alpha$ is the $\alpha$-quantile of a $N(0,1)$ random variable. For small sample sizes, the distribution of the sample eigenvalues is not symmetric: this is illustrated by Figure 21. Note also that Theorem 4.2.16 requires assuming that $\lambda_i \neq \lambda_j$ for $i \neq j$, but these are precisely the values we are trying to estimate. Furthermore, notice that important quantities in the variance of the sample eigenvector are the *eigengaps* $\lambda_i - \lambda_k, k \neq i$, and in particular, if $\lambda_{i+1} = \lambda_i$, then the variance is infinite. This is because when equality of the eigenvalues occurs, the corresponding eigenvectors are not uniquely defined! Also, when eigenvalues are equal, the sample eigenvalues are close but not equal. Both of these are illustrated in Figure 22, where we see that when the eigengap becomes smaller, the variance of the sample eigenvectors increases, and it becomes infinite when the eigengap is zero: the distribution of the first sample eigenvector becomes uniform on the circle.

**Example 4.2.17** (Code for quantifying the finite sample distribution of sample eigenvectors and eigenvalues)**.** The code used to get Figures 21 is given below.

---

```
rot = matrix(c( 1,1, -1 , 1), ncol=2)/sqrt(2)

set.seed(1)
sd_x <- 2
n <- 10
Nrep = 5e4
rep_eigenvalues0 <- array(NA, c(Nrep, 2))
rep_eigenvectors0 <- array(NA, c(Nrep, 2, 2))

for( i in 1:Nrep ){
    x = rnorm(n, sd=sd_x)
    y = rnorm(n)
```

```
    X00 =  cbind(x,y) %*% t(rot)
    X00_pca <- prcomp(X00)
    rep_eigenvectors0[i,,] <- X00_pca$rotation
    rep_eigenvalues0[i,] <- X00_pca$sdev^2
}


op <- par(mfrow=c(2,1), mai=c(.4,0.4,1,0.1))
x <- rep_eigenvalues0[,1]
hist(x, 'FD', freq=FALSE, col=1,
     main='Distribution of the variance of the 1st PC')
tt <- seq(0,10, len=1e4)
lines(tt, dnorm(tt, mean=sd_x^2, sd=sqrt(2/n)*sd_x^2), col=2, lwd=2)
#lines(tt, dnorm(tt, mean=sd_x^2, sd=sd(x)), col=2, lwd=2)
abline(v=sd_x^2, col=2, lwd=2, lty=2)
#    lines(bhist£breaks, c(bhist£density,0), type='s', col=3)
#
x <- rep_eigenvalues0[,1]/(rowSums(rep_eigenvalues0))
hist(x, 'FD', col=1, freq=FALSE,
     main='Distribution of % of variance explained by 1st PC')
lines(tt, dnorm(tt, mean=mean(x), sd=sd(x)), col=2, lwd=2)
abline(v=sd_x^2/(sd_x^2 + 1), col=2, lwd=2, lty=2)
#    lines(bhist2£breaks, c(bhist2£density,0), type='s', col=3)
par(op)
```

Here is the code for Figure 22.

```
rot = matrix(c( 1,1, -1 , 1), ncol=2)/sqrt(2)
plot_vector <- function(v, alpha=1, ...){
    lines(alpha*matrix(c(-v[1], v[1], -v[2], v[2]), 2,2), ...)
}

n <- 50
Nrep = 1e3

sd_seq <- c(2,1.4,1)
rep_eigenvalues <- array(NA, c(length(sd_seq), Nrep, 2))
rep_eigenvectors <- array(NA, c(length(sd_seq), Nrep, 2, 2))

set.seed(1)
for(sd_i in seq(along=sd_seq))
{
    sd_x <- sd_seq[sd_i]
    for( i in 1:Nrep ){
        x = rnorm(n, sd=sd_x)
        y = rnorm(n)
```

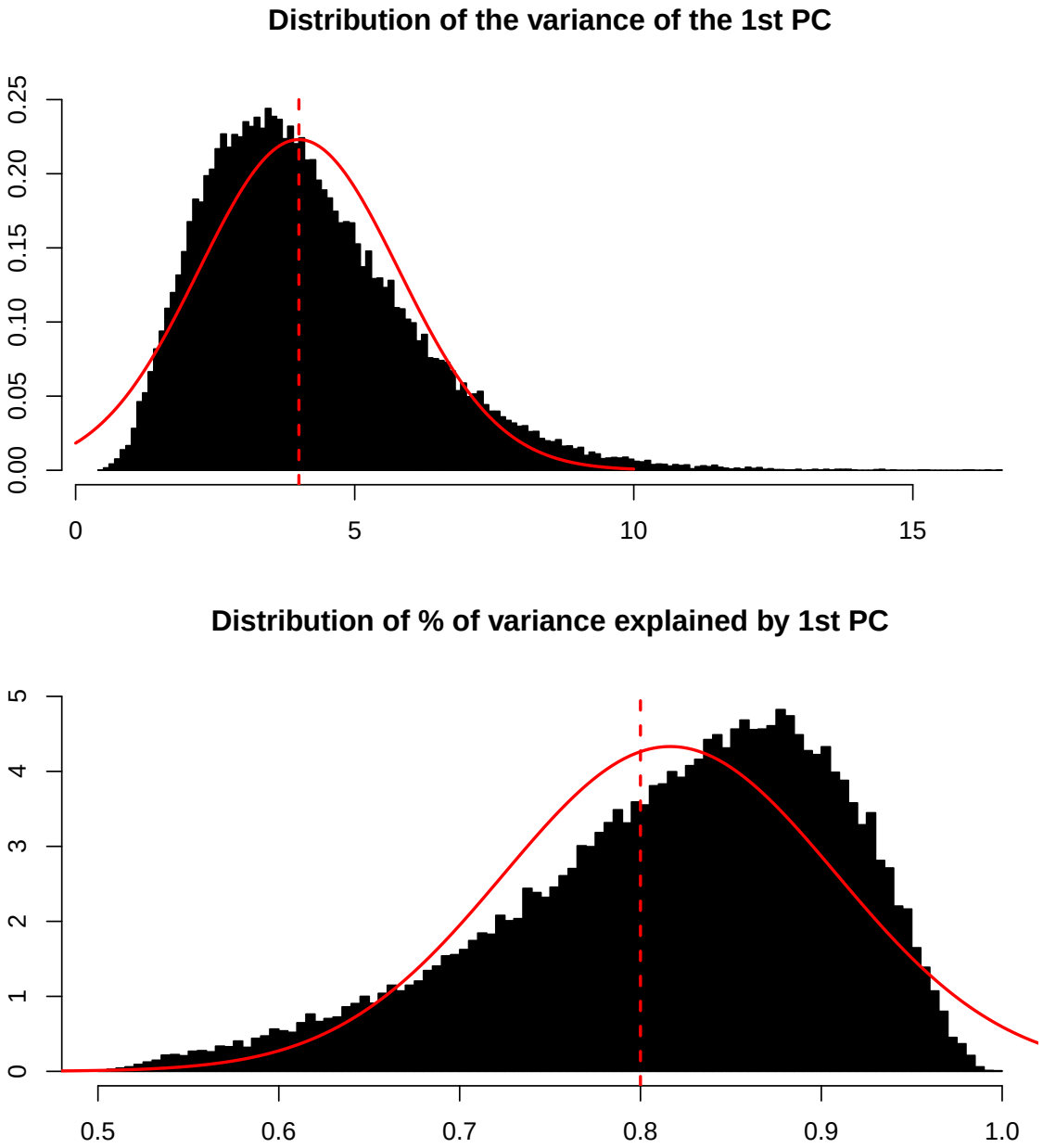



Figure 21: Sample distribution of the variance explained by the first PC (top) and sample distribution of the percentage of variance explained by the first PC (bottom). The distributions have been computed using $N = 5 \cdot 10^4$ replicates of a sample of size 10. Overlaid are the Gaussian density estimate (notice in particular in the top figure, the Gaussian gives non-zero probability for negative values...). The true parameter is denoted by the red vertical dashed line. Notice that the distributions are asymmetric. This asymmetry goes away for larger sample sizes (this cannot be seen in this Figure).

```
        X0 =  cbind(x,y) %*% t(rot)
        X0_pca <- prcomp(X0)
        rep_eigenvectors[sd_i, i,,] <- X0_pca$rotation
        rep_eigenvalues[sd_i, i,] <- X0_pca$sdev^2
    }
}

op <- par(mfrow=c(3,2))
for(sd_i in seq(along=sd_seq)){
    par(mai=c(0.1,0.0,0.1,0))
    #
    plot(X0, asp=1, xlim=c(-5,5), ylim=c(-5,5), axes=TRUE, xaxt='n',
         yaxt='n', type='n')
    for(i in 1:min(Nrep,5e2)){
        plot_vector(rep_eigenvectors[sd_i, i,,1], alpha=10, lwd=.1,
                    col=gray(0, alpha=.2))
    }
    if(sd_seq[sd_i] != 1) ## plot true eigenvector if well defined
    plot_vector(rot[,1], alpha=10, lwd=2, col=1, lty=2)
#
    h1 <- hist(rep_eigenvalues[sd_i, ,1], 'FD', plot=FALSE)
    h2 <- hist(rep_eigenvalues[sd_i, ,2], plot=FALSE, 'FD')
    ylim <- range(c(h1$density, h2$density))
    xlim <- range(c(h1$breaks, h2$breaks))
    par(mai=c(0.3,0.3,0.1,0))
    plot(h1$breaks, c(h1$density, 0), col=1, xlim=xlim, ylim=ylim,
         type='s', bty='n')
    abline(v=sd_seq[sd_i]^2, col=1, lwd=2, lty=2)
    lines(h2$breaks, c(h2$density,0), type='s', col=2)
    abline(v=1, col=2, lwd=2, lty=2)
}
par(op)
```

### 4.2.9 Principal components plots (EDA)

A plot of the PC scores (or commonly known as *principal components plot*) gives a low-dimensional graphical representation of the data. This serves many purposes, for instance by allowing one to get an idea of the (Euclidean) distances between different observations, helps identifying outliers, allows one to see clusters in the data, and if labels are available, allows one to establish how easy (or difficult) a classification problem might be. All of these can also give insight into formulating a model for the data.

The PC scores are obtained simply by projecting $\mathtt{X}$, the $n \times p$ matrix containing the observed data, on the first few eigenvectors. Letting $E = (\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_p)$, we can compute

$$\mathtt{Y} = \mathtt{X}E,$$

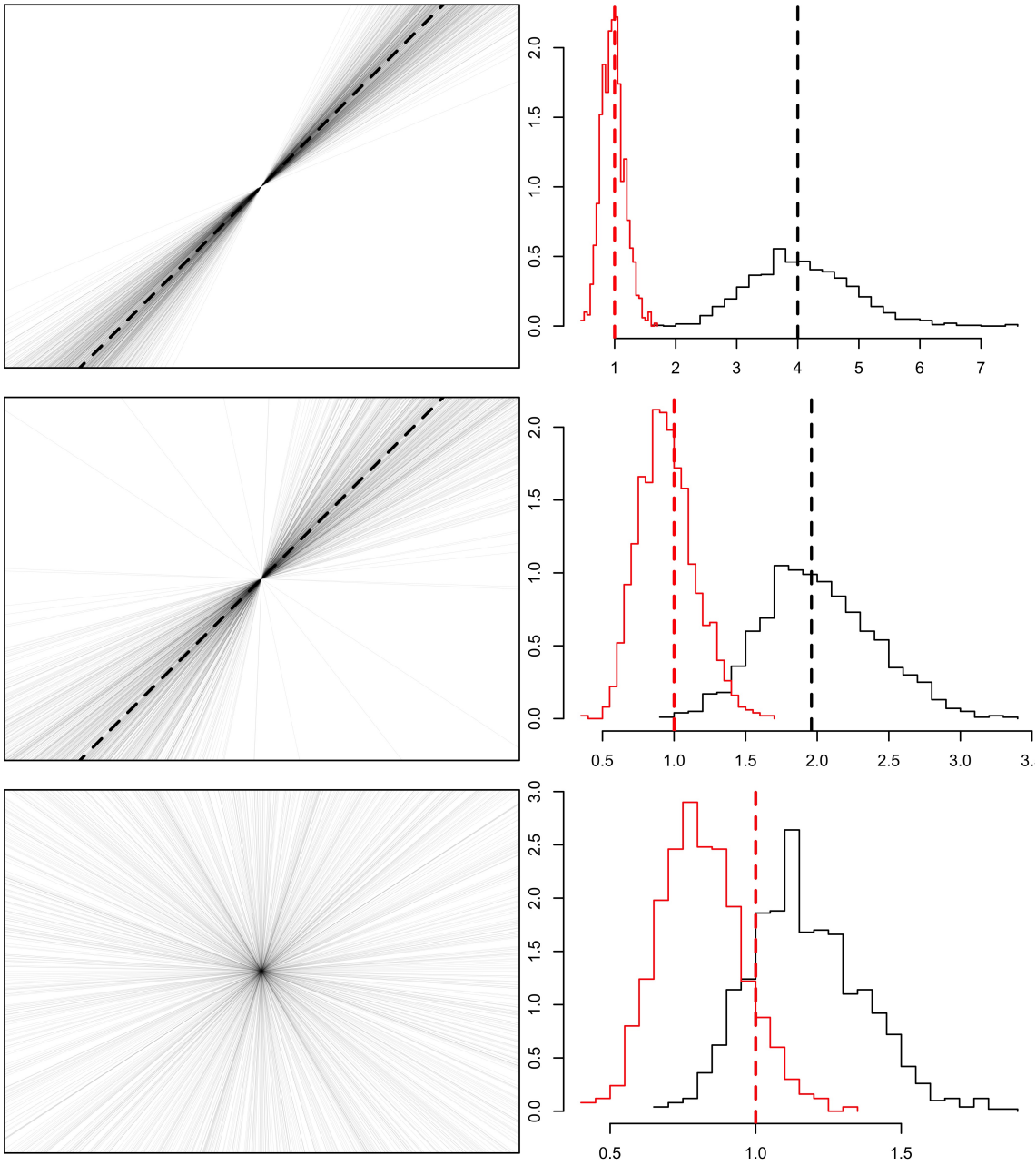



Figure 22: Empirical distribution of eigenvectors and eigenvalues of the sample covariance. Each row corresponds to a covariance with a specific eigengap (the difference between first and second eigenvalue), and the difference between the rows is only in the eigenvalues of the covariance. The true eigenvalues (unknown) are depicted by dashed vertical lines in the right-hand side plot (black for largest eigenvalue, red for 2nd largest eigenvalue). Overlayed is a histogram of the distribution of the sample eigenvalues, in the corresponding color, obtained by simulating $10^3$ replicates of a sample size 50 dataset. On the left are plotted in thin lines the first PC loading for half of the $10^3$ replicates. Overlaid as a thick black line is the true 1st eigenvector of the covariance. In the bottom row, the first two eigenvalues are equal, and hence the true 1st eigenvector is not uniquely defined.

which is the $n \times p$ matrix whose columns contain the PC scores. Plotting column $k$ versus column $l$ of `Y` gives a 2D scatterplot of the PC scores $k$ and $l$ of the data `X`.

**Example 4.2.18** (cars dataset PCA)**.** Consider the cars dataset, which measures 7 continuous characteristics for 32 cars. Below we load the data, standardize all variables to 0 mean and variance 1, and obtain the eigendecomposition of the covariance matrix (equivalently, the correlation matrix as the data are standardized). Two components explain roughly 90% of the variance, see R code below), hence we plot the two first principal components and add labels to identify each car.

```
require(grDevices)
op <- options(digits = 2) # print only 2 digits
data(mtcars)
nn <- rownames(mtcars)
## column centering and scaling to have variance 1 in each column
z <- scale(mtcars[,1:7], center=TRUE, scale=TRUE)
## check:
colMeans(z)
```

```
##      mpg      cyl     disp       hp     drat       wt     qsec
##  7.1e-17 -1.5e-17 -9.1e-17  1.0e-17 -2.9e-16  4.7e-17  5.3e-16
```

```
S <- cov(z)
diag(S)
```

```
##  mpg  cyl disp   hp drat   wt qsec
##    1    1    1    1    1    1    1
```

```
l <- eigen(S)$values
v <- eigen(S)$vector

print(l) # the eigenvalues or percentage of variance for each PC
```

```
## [1] 5.086 1.157 0.345 0.158 0.129 0.076 0.049
```

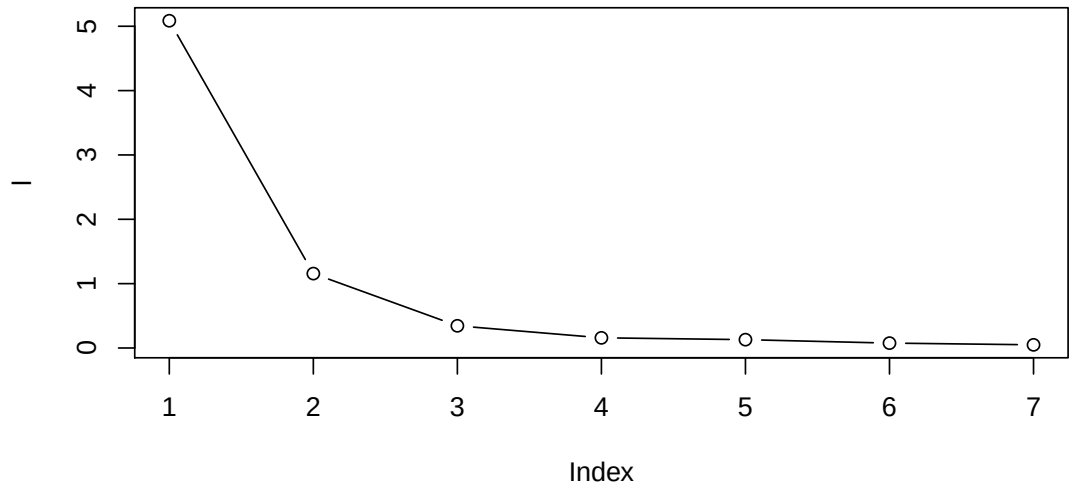
```
print(100*cumsum(l)/sum(l)) # the cumulative percentage of variance explained
```

```
## [1]  73  89  94  96  98  99 100
```

```
plot(l,type='b') # scree-plot

print(v) # the principal component scores (the columns of v)
```

```
##       [,1]   [,2]  [,3]    [,4]  [,5]   [,6]   [,7]
## [1,]  0.41 -0.083  0.24  0.7668 -0.21 -0.090  0.351
## [2,] -0.42 -0.078  0.19  0.1939  0.24  0.781  0.273
## [3,] -0.42  0.082 -0.12  0.5877  0.15 -0.162 -0.638
## [4,] -0.39 -0.337 -0.20 -0.0069 -0.83  0.046  0.039
## [5,]  0.33 -0.449 -0.76  0.1171  0.22  0.233 -0.037
## [6,] -0.39  0.322 -0.44  0.1073  0.17 -0.366  0.613
## [7,]  0.24  0.749 -0.29  0.0614 -0.33  0.407 -0.131
```

```
options(op)
```




```
pc <- z %*% v

xlim <- range(pc[,1:2])
op <- par()
layout(c(1,1,2))
plot(pc[,1:2],xlab='PC 1',ylab='PC 2',xlim=xlim,ylim=xlim,
                          main="PC Scores 1 vs 2")
text(pc[,1:2],nn,pos=3)

#BOOTSTRAP (NON-EXAMINABLE)
B <- 10^4
lb <- matrix(NA,nrow=B,ncol=ncol(z))
for (b in 1:B) {
  zb <- z[sample(1:nrow(z),replace=TRUE),]
  Sb <- cov(zb)
  lb[b,] <- eigen(Sb)$values
}
propvar <- rowSums(lb[,1:2])/rowSums(lb)
quantile(propvar,probs=c(.025,.975))

##      2.5%     97.5%
## 0.8718952 0.9259666

hist(propvar, main='Bootstrap samples',
xlab='Proportion explained variance by Components 1-2', cex.lab=1.3, 'fd')
par(op)
```

The resulting principal components plot is in Figure 23 (top) We do not detect any obvious outliers in the plot, and we observe that cars from the same manu-

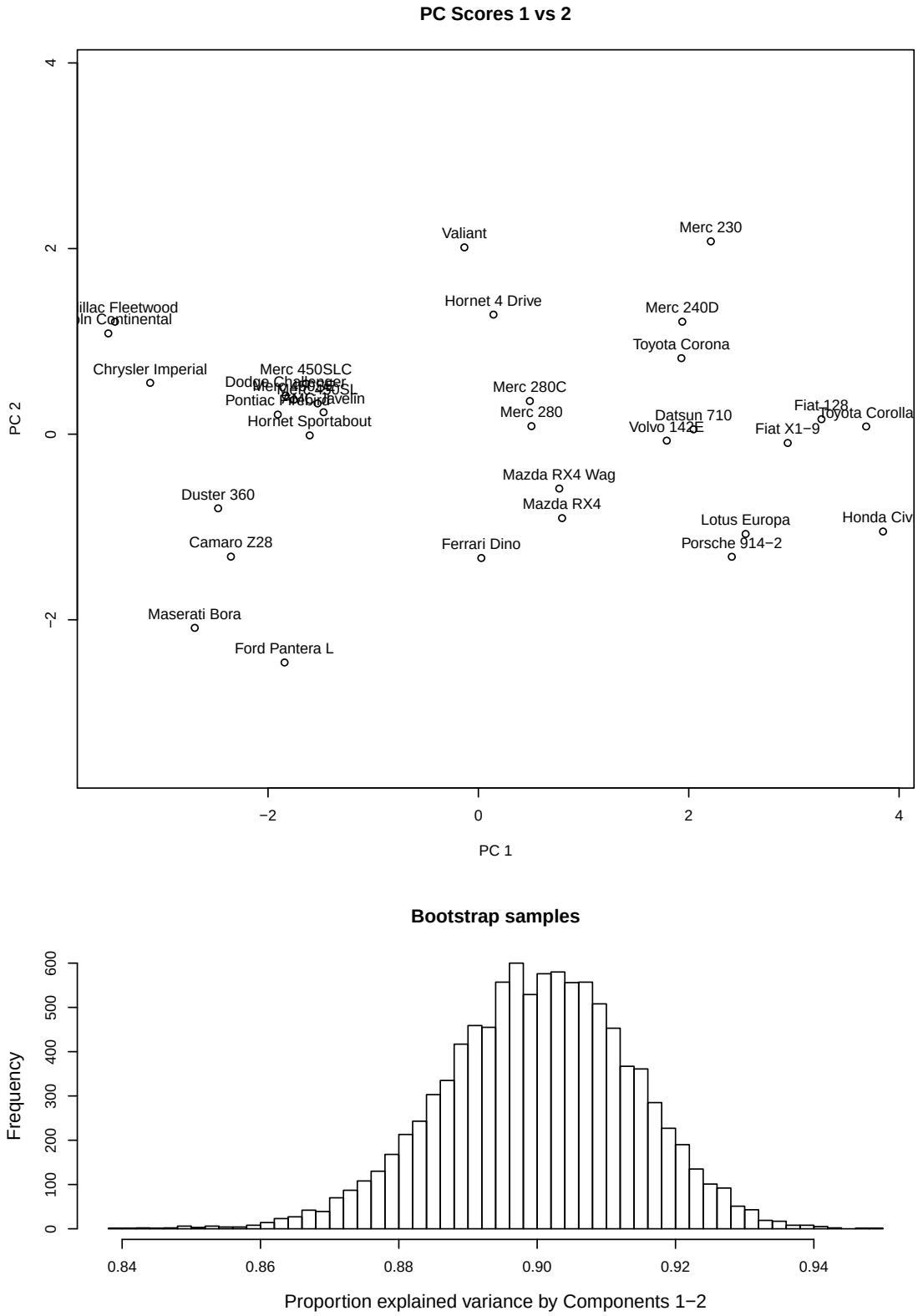



Figure 23: PC Scores 1-2 and bootstrap histrogram for the percentage of variance explained by first 2 PCs.

facturer often appear nearby in the plot, hence suggesting that they had similar values for the 7 recorded characteristics. The bottom panel shows a histogram of the bootstrap samples of $\sum_{i=1}^{2} \hat{\lambda}_i^{(b)} / \sum_{i=1}^{p} \hat{\lambda}_i^{(b)}$. The histogram is roughly centred on the 89.2% in the observed data, but also characterizes the uncertainty around this number.

---

**Example 4.2.19** (Zip code PCA)**.** We go back to our `ZIP code` example. We now perform a PCA on the dataset. We plot in Figure 24 the PC scores 1 versus 2, where each point (observation) has been given a colour corresponding to the digit it represents. Notice how some digits are well separated, while others or not. We can go beyond just looking at the first 2 PCs, and do a pairs plot of the PC scores, see Figure 26. To interpret the PC scores, we can look at the PC loadings, given in Figure 27. We see for instance that PC score 1 is given by the difference between a weighted average around the center, and a weighted average in the center of the image (this is the exact interpretation). The shape of the positive values resembles a "0", and the shape of the negative values resembles a "1", so one would expect to see that the digits "1" have low PC 1 values, and digits "0" have high PC 1 values. An inspection of Figure 24 shows that this is indeed the case. The screeplot for the Zip code dataset is shown in Figure 25. To get the approximate interpretation of PC scores, look at Figures 28, 29, 30, and 31, which show the effect of negative and positive scores for PCs 1, 2, 3, and 4, respectively. We can also look at how well individual observations are approximated by using only the first $K$ principal components. Figure 32 show this for an image of the digit "9" for $K = 1, \ldots, 15$. Figure 33 show this for an image of the digit "4" for $K = 1, \ldots, 41$. We notice that about 6 PCs are enough to recognise the digit in the first case, but we need up to about 24 PCs to recognise the digit in the second case. There is no clear shoulder in the screeplot, but we need 39 PCs to get at least 90% of the variance explained.

---

```
## Loading required package:  spam
## Loading required package:  dotCall64
## Loading required package:  grid
## Spam version 2.2-0 (2018-06-19) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g.  'help( chol.spam)'.
##
## Attaching package:  'spam'
## The following objects are masked from 'package:base':
##
##    backsolve, forwardsolve
## Loading required package:  maps
## See www.image.ucar.edu/~nychka/Fields for
```

```
##  a vignette and other supplements.
```

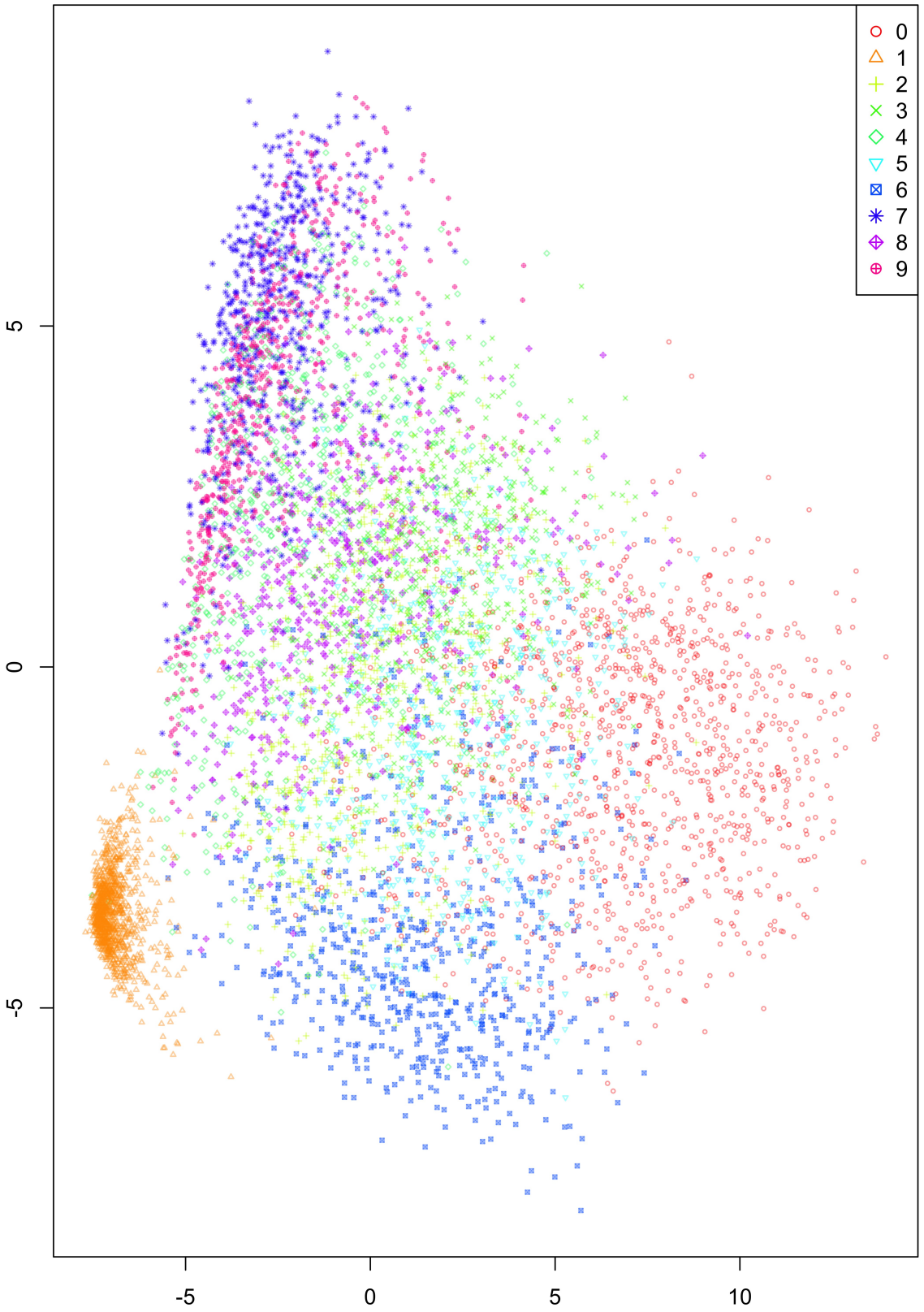



Figure 24: PC scores 1 versus 2 for the Zip codes. Notice that some digits are very well separated, such as 0 and 1, while other are not so clearly separated, such as 3 and 5.

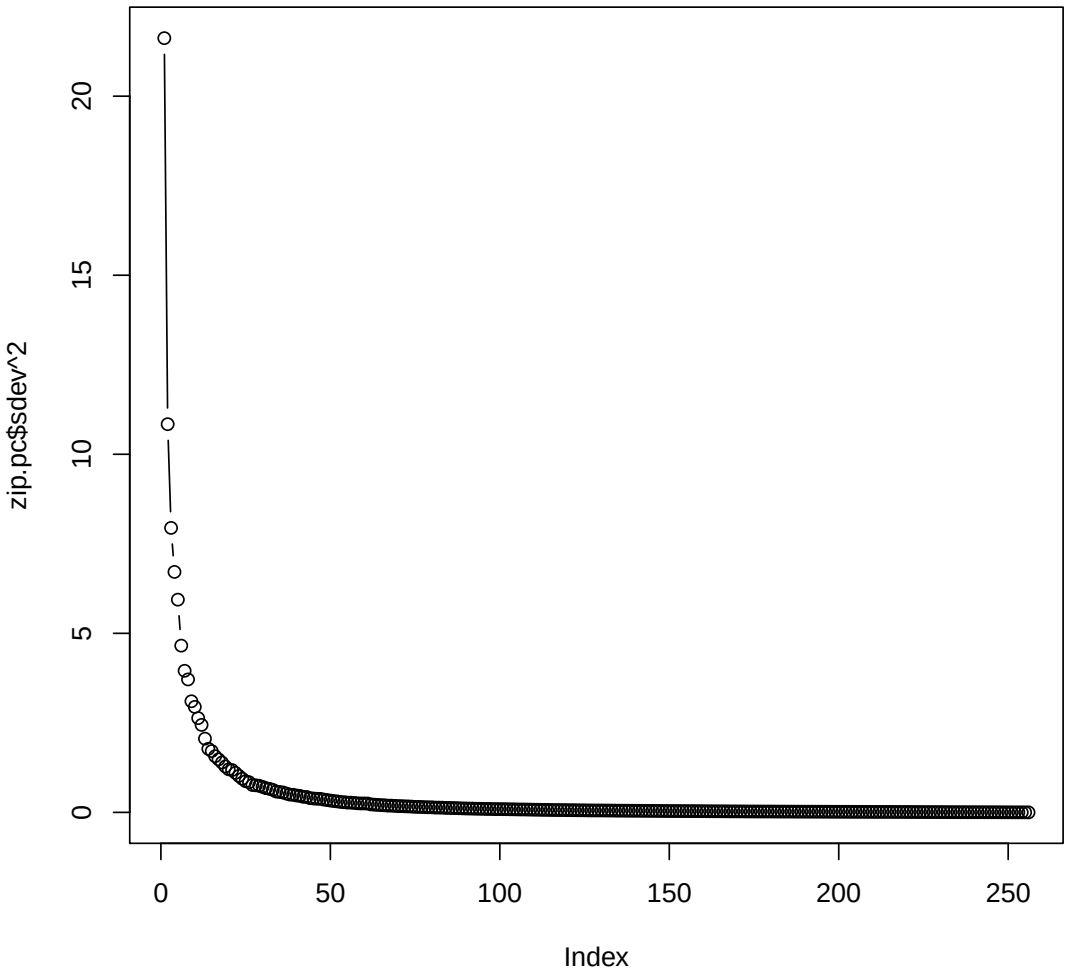



Figure 25: Screeplot for the Zip code dataset.

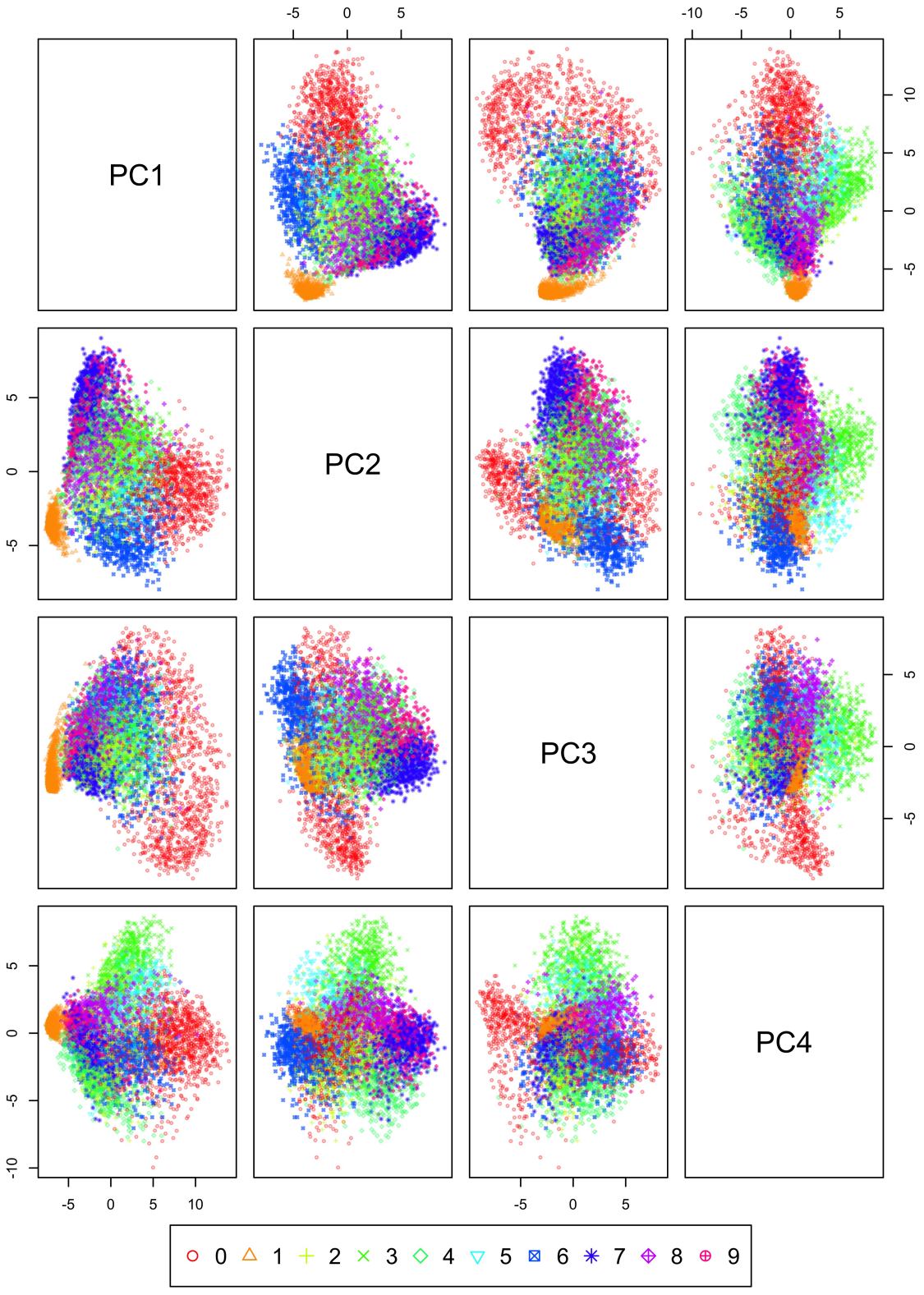



Figure 26: Pairs plot of the PC scores for the Zip codes data. We now see for instance that PC 4 is reasonably good at distinguishing digits 3 and 4. This wasn't the case with PCs 1 and 2.

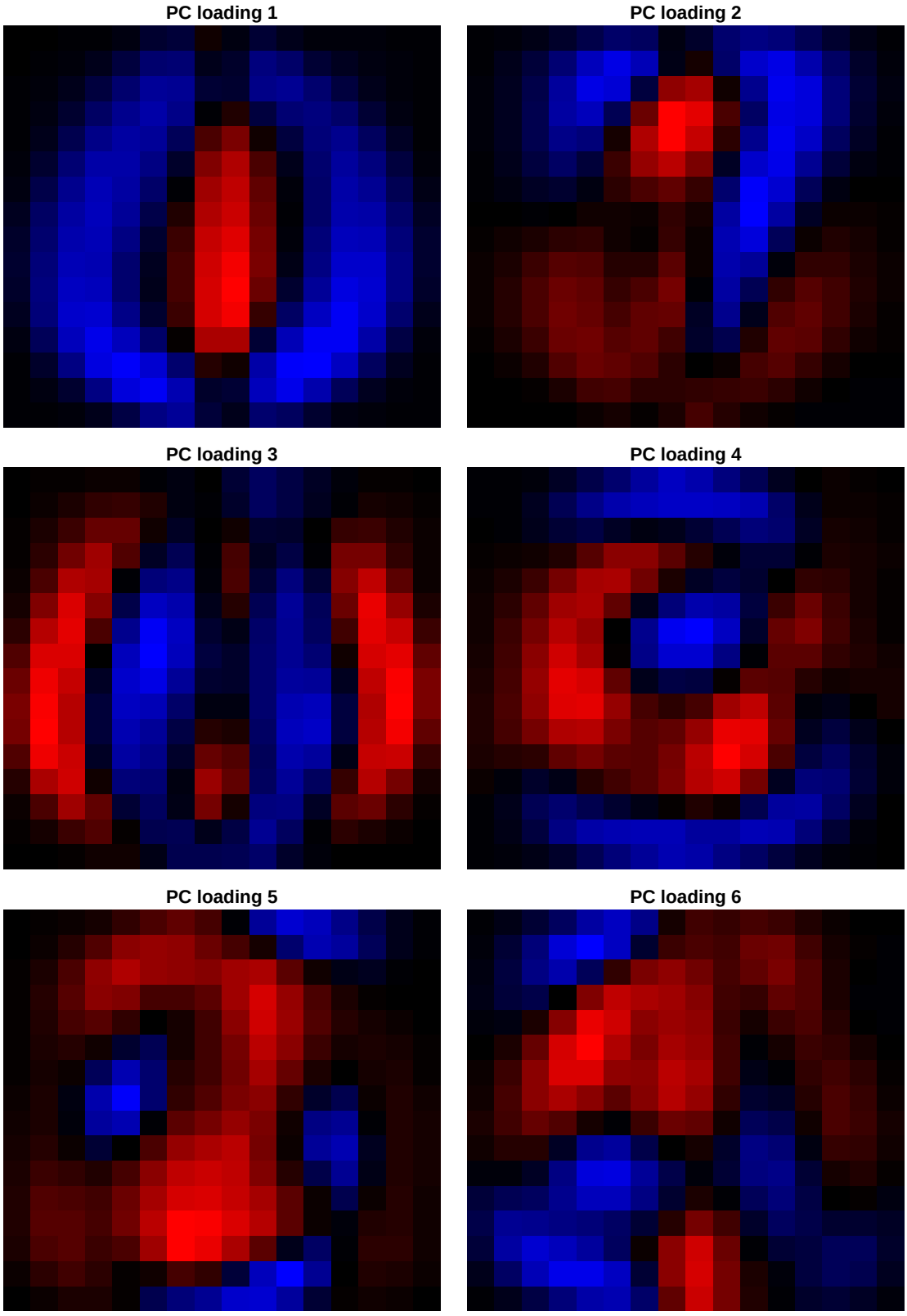



Figure 27: PC Loadings of the ZIP CODE dataset. Red denotes negative values, blue positive values. Notice that each PC loading is a $16 \times 16$ image.

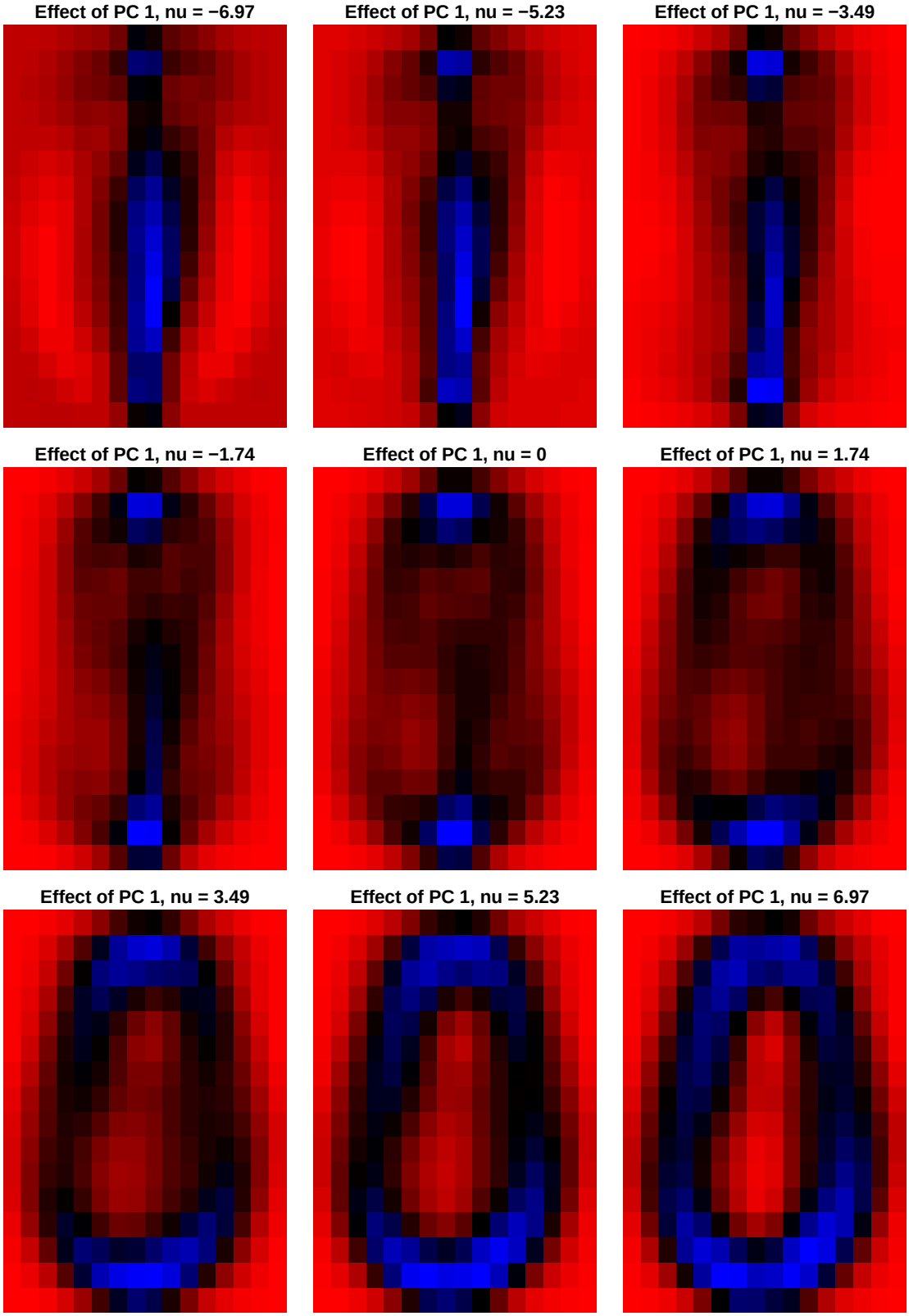



Figure 28: Effect of PC Loading 1 of the ZIP CODE dataset. Each subplot show the image for $\overline{\mathbf{x}} + \nu * \mathbf{e}_1$. The case $\nu = 0$ is just the (sample) mean digit. Red denotes negative values, blue positive values.

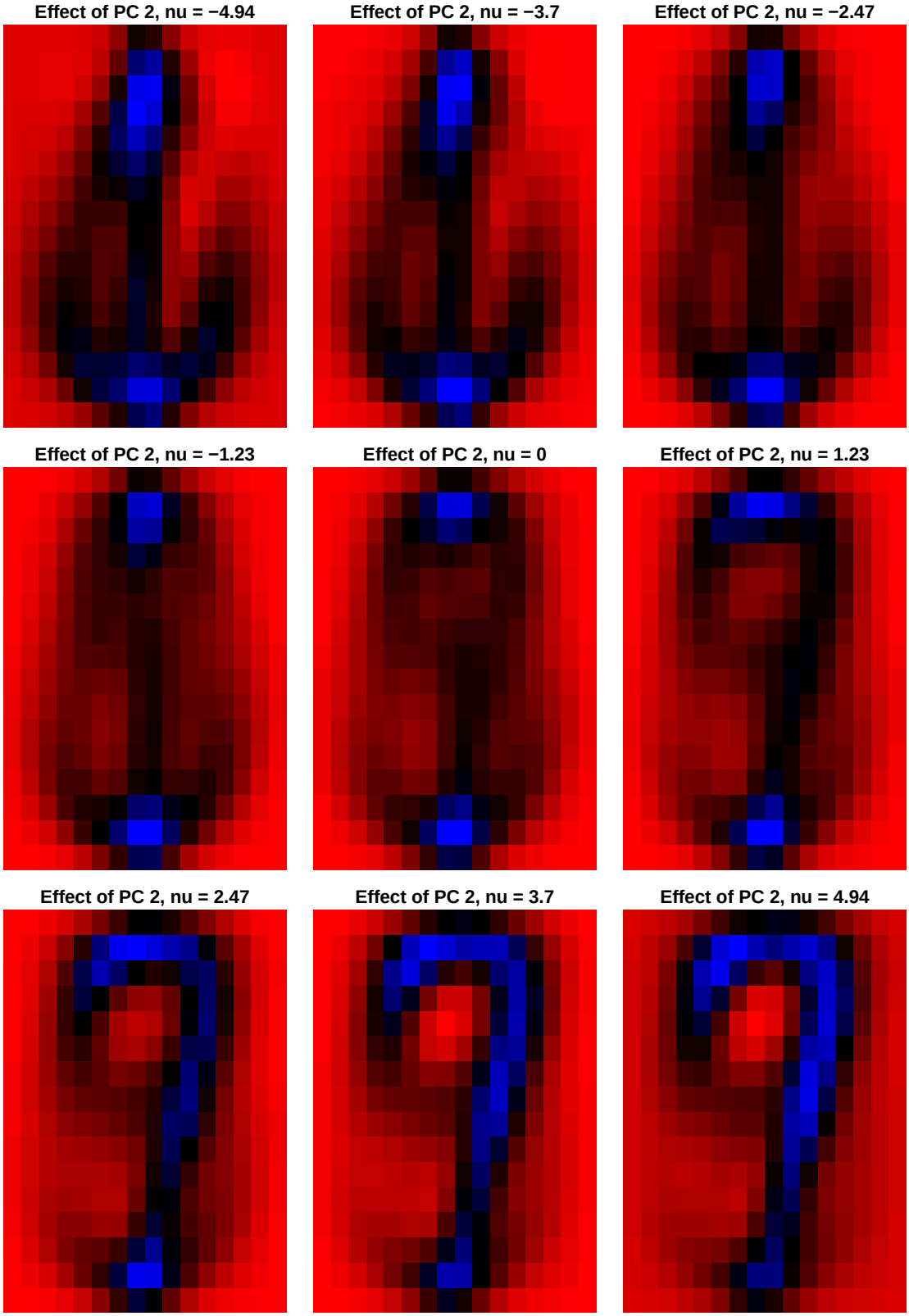



Figure 29: Effect of PC Loading 2 of the ZIP CODE dataset. Each subplot show the image for $\overline{\mathbf{x}} + \nu * \mathbf{e}_2$. The case $\nu = 0$ is just the (sample) mean digit. Red denotes negative values, blue positive values.

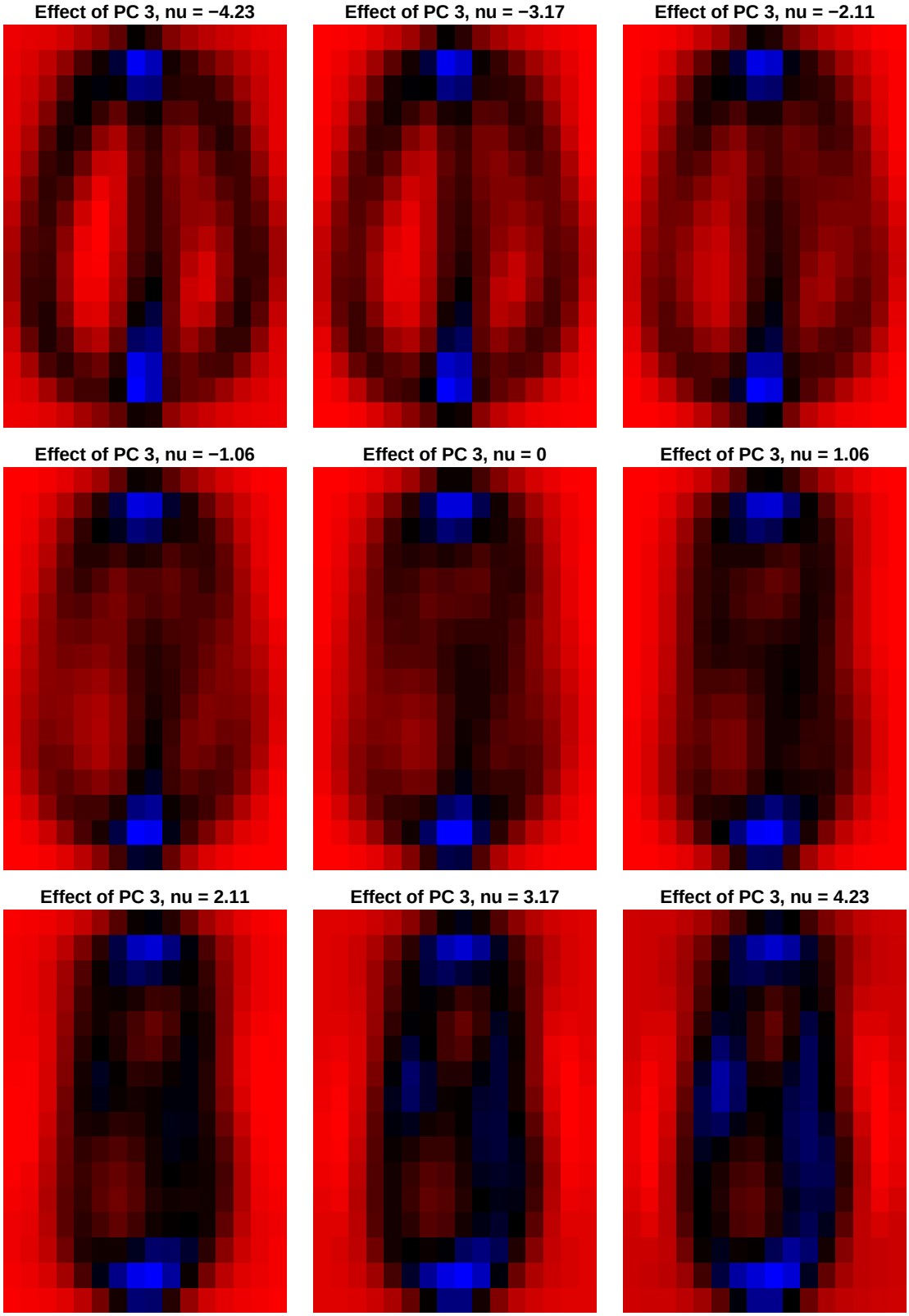



Figure 30: Effect of PC Loading 3 of the ZIP CODE dataset. Each subplot show the image for $\overline{\mathbf{x}} + \nu * \mathbf{e}_3$. The case $\nu = 0$ is just the (sample) mean digit. Red denotes negative values, blue positive values.

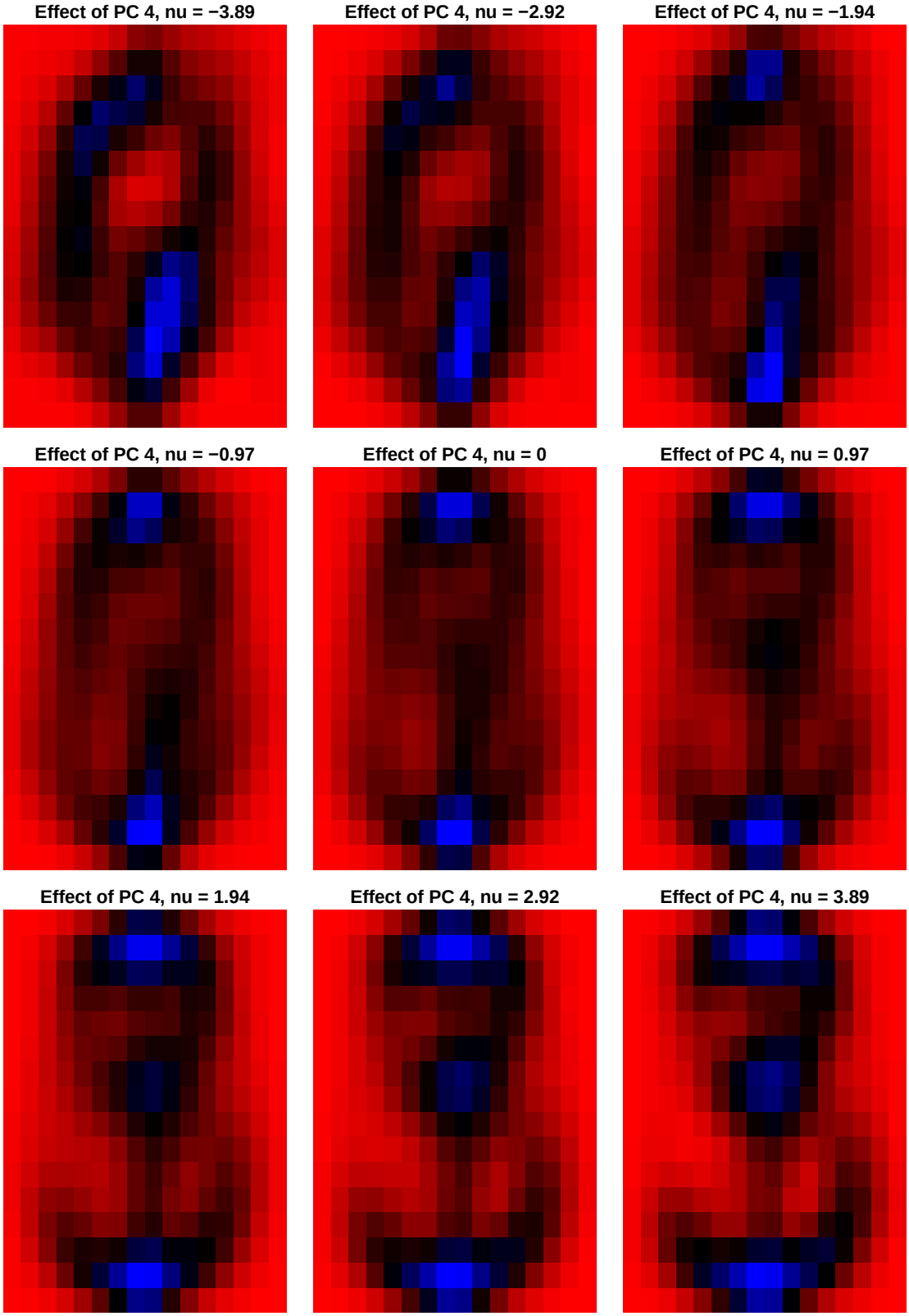



Figure 31: Effect of PC Loading 4 of the ZIP CODE dataset. Each subplot show the image for $\overline{\mathbf{x}} + \nu * \mathbf{e}_4$. The case $\nu = 0$ is just the (sample) mean digit. Red denotes negative values, blue positive values.

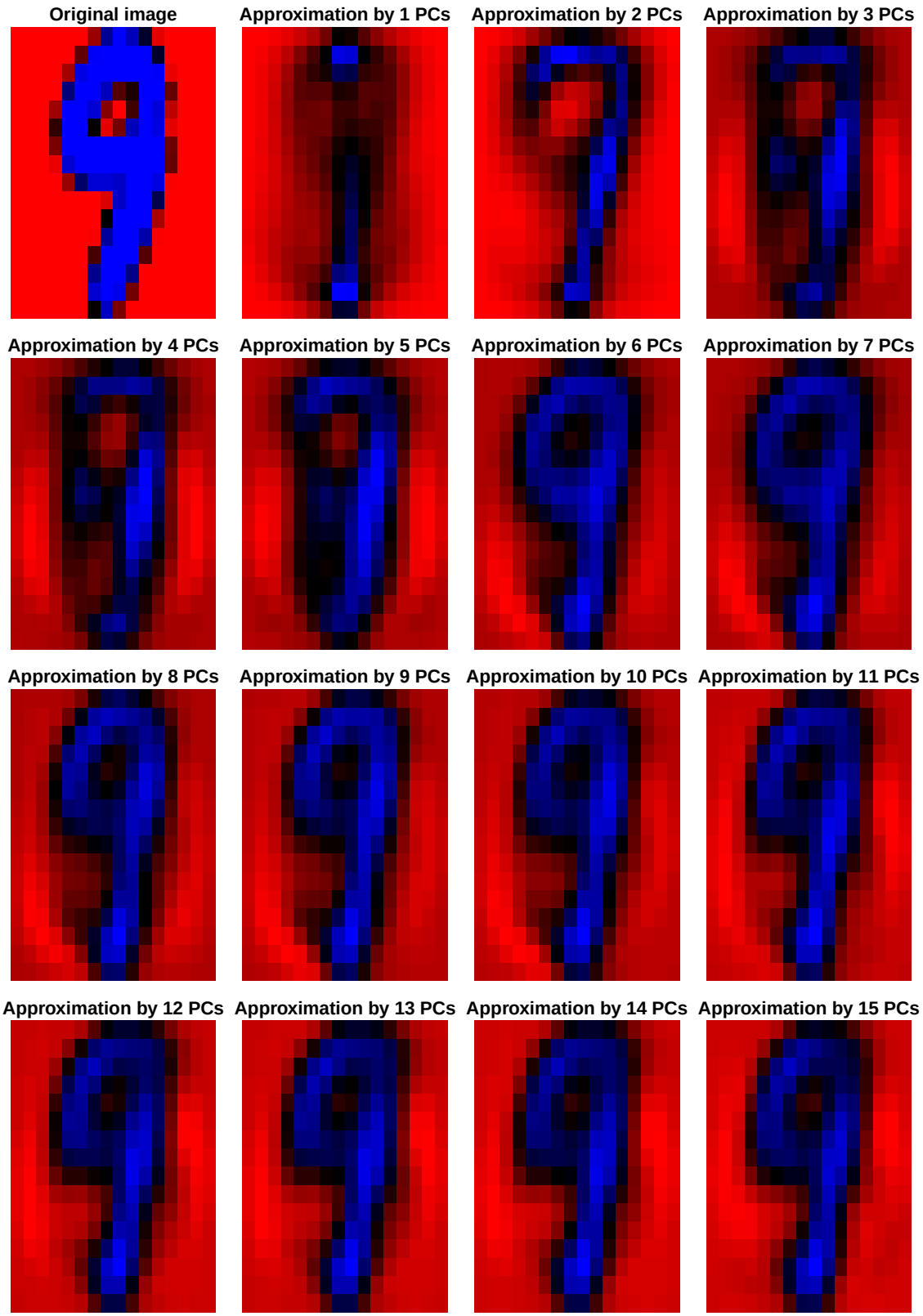



Figure 32: Approximations of observation $i = 332$, using $k = 1, 2, \ldots, 15$ PCs. The top-left subplot is the original observation (the $i$-th row of $\mathtt{X}$). Each following subplot is the $i$-th row $\mathtt{X}_k$. Notice that we can recognise the digit with 6 PCs already. Red denotes negative values, blue positive values.

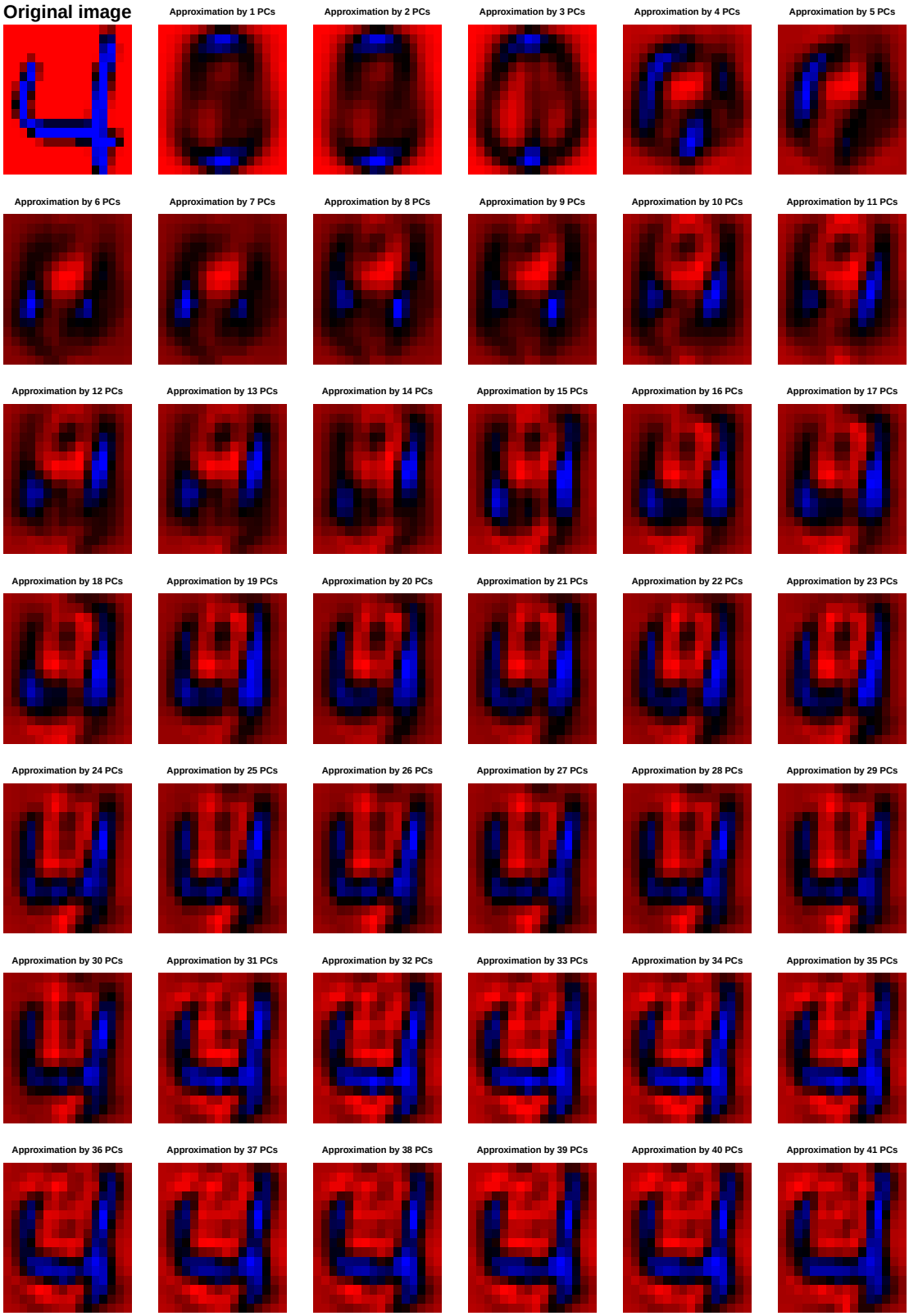



Figure 33: Approximations of observation $i = 1111$, using $k = 1, 2, \ldots, 41$ PCs. The top-left subplot is the original observation (the $i$-th row of $\mathtt{X}$). Each following subplot is the $i$-th row $\mathtt{X}_k$. Notice that we can recognise the digit only with about 31 or more PCs. Red denotes negative values, blue positive values.

## 4.3 The Biplot

The biplot is a graphical tool that *jointly represents* rows and columns of a data matrix $\mathtt{X}$ with $n$ rows and $p$ columns. It is directly related to principal components analysis, but unlike with principal components we can include individuals (rows) and variables (columns) in a single plot. Without loss of generality suppose that the columns in $\mathtt{X}$ are centred at 0, otherwise we replace $\mathtt{X}$ by $H\mathtt{X}$, where $H$ is the $n \times n$ centering matrix. Recall that the sample covariance matrix can be expressed as $S = \frac{1}{n-1}\mathtt{X}^\mathsf{T}\mathtt{X}$ (no $H$ here since the data matrix is already column-centered).

The biplot is based on the *singular value decomposition* of $\mathtt{X}$, which has already been introduced in Section 2.2. Let $\mathtt{X} = ULV^\mathsf{T} = \sum_{j=1}^{q} l_{jj}\mathbf{u}_j\mathbf{v}_j^\mathsf{T}$ be an SVD of $\mathtt{X}$. Being quite informal with the notation, writing $\mathtt{X} = (UL^{1/2})(L^{1/2}V^\mathsf{T})$, we notice that $(\mathtt{X})_{ij}$ can be expressed as an inner-product between the $i$-th row of $UL^{1/2}$ and $j$-th column of $L^{1/2}V^\mathsf{T}$. If we could plot (as vectors) these rows and columns, then we could "read" the value of $(\mathtt{X})_{ij}$ from the plot. However, the rows and columns we are talking about are vectors in $\mathbb{R}^q$, where $q$ is the rank of $\mathtt{X}$. Recall that by the Young–Eckart–Mirsky Theorem (Theorem 2.2.3), the best rank $k$ approximation of $\mathtt{X}$ is given by $\mathtt{X}_k = \sum_{j=1}^{k} l_{jj}\mathbf{u}_j\mathbf{v}_j^\mathsf{T}$. Now if we take $k = 2$ or $k = 3$, we could plot the aforementioned rows and columns in the same plot, and use this plot to "read" the entries of $\mathtt{X}$. This is the idea behind the biplot:

**Definition 4.3.1 (The Biplot).** We approximate the $n \times p$ column centered data matrix $\mathtt{X}$ using the first $k$ singular vectors:

$$\begin{aligned}
\mathtt{X} \approx \mathtt{X}_k &= \sum_{j=1}^{k} l_{jj}\mathbf{u}_j\mathbf{v}_j^\mathsf{T} = \begin{pmatrix}\mathbf{u}_1 & \dots & \mathbf{u}_k\end{pmatrix}\begin{pmatrix} l_{11} & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & l_{kk}\end{pmatrix}\begin{pmatrix}\mathbf{v}_1^\mathsf{T} \\ \dots \\ \mathbf{v}_k^\mathsf{T}\end{pmatrix} \\
&= \begin{pmatrix} u_{11} & \dots & u_{k1} \\ \vdots & & \vdots \\ u_{1n} & & \dots u_{kn}\end{pmatrix}\begin{pmatrix} l_{11} & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & l_{kk}\end{pmatrix}\begin{pmatrix} v_{11} & \dots & v_{1p} \\ \vdots & & \vdots \\ v_{k1} & \dots & v_{kp}\end{pmatrix} \\
&= \begin{pmatrix} u_{11}\sqrt{l_{11}} & \dots & u_{k1}\sqrt{l_{kk}} \\ \vdots & & \vdots \\ u_{1n}\sqrt{l_{11}} & \dots & u_{kn}\sqrt{l_{kk}}\end{pmatrix}\begin{pmatrix} v_{11}\sqrt{l_{11}} & \dots & v_{1p}\sqrt{l_{11}} \\ \vdots & & \vdots \\ v_{k1}\sqrt{l_{kk}} & \dots & v_{kp}\sqrt{l_{kk}}\end{pmatrix} = \tilde{U}_k\tilde{V}_k^\mathsf{T}.
\end{aligned}$$

Then the coordinates of the rows in $\mathtt{X}$ (the observations) in the *$k$-dimensional biplot* are given by the rows in the $n \times k$ matrix $\tilde{U}_k$, and the coordinates of the columns of $\mathtt{X}$ (the variables) are given by the rows in the $p \times k$ matrix $\tilde{V}_k$. **Usually, we take $k = 2$, as 2-dimensional plots are easy to visualize**.

As an important remark, Definition 4.3.1 is based on splitting the $l_{ii}$s evenly between $\tilde{U}$ and $\tilde{V}$. Nothing stops us from using the alternative approximation $X \approx U\tilde{V}^\mathsf{T}$ where now $\tilde{V}^\mathsf{T}$ contains $l_{ii}$ instead of $\sqrt{l_{ii}}$. This alternative emphasizes relations between columns of $X$. Similarly, we could define $X \approx \tilde{U}V^\mathsf{T}$ where $\tilde{U}$ contains $l_{ii}$s, which emphasizes relations between rows of $X$. All these alternatives result in

exactly the same coordinates up to a scale factor given by $l_{ii}$, i.e. the biplot looks similar except that either rows or columns are emphasised by being more spread out. Different software implementations may produce different versions of biplots, depending on their emphasis on rows, columns or neither. For our purposes, we will stick to the choice in Definition 4.3.1, as it emphasizes equally the representation of rows and columns.

**Example 4.3.2** (toy data Biplot)**.** Consider the data matrix

$$\begin{pmatrix} -1 & -1 \\ 0 & 0.2 \\ 1 & 1 \end{pmatrix}$$

with 2 variables and 3 individuals. Its column-centered version is

$$\mathtt{X} = \begin{pmatrix} -1 & -1.07 \\ 0 & 0.13 \\ 1 & 0.93 \end{pmatrix}$$

and is plotted in Figure 34 (left). Below is the code to compute and plot `X` on a 2-dimensional biplot. The matrices $\tilde{U}, \tilde{V}$ contain as rows the coordinates of the observations, respectively the variables, on a 2-dimensional biplot, and are computed using the SVD of `X` using the following code:

```
X <- matrix(c(-1,-1,0,0.2,1,1),ncol=2,byrow=TRUE) # the data matrix
X <- scale(X, center=TRUE, scale=FALSE) # column centered data matrix

(X.svd <- svd(X)) # computing the SVD of X

## $d
## [1] 2.0033417 0.1152774
##
## $u
##              [,1]       [,2]
## [1,] -0.72953259  0.3666727
## [2,]  0.04721843 -0.8151301
## [3,]  0.68231416  0.4484574
##
## $v
##           [,1]       [,2]
## [1,] 0.7047459  0.7094598
## [2,] 0.7094598 -0.7047459

X.svd$u %*% diag(X.svd$d) %*% t(X.svd$v) ## recover X using its SVD

##      [,1]       [,2]
## [1,]   -1 -1.0666667
## [2,]    0  0.1333333
## [3,]    1  0.9333333
```

```
(Utilde <- X.svd$u %*% diag(sqrt(X.svd$d)) ) # coordinates of the observations

##              [,1]       [,2]
## [1,] -1.03257643  0.1244947
## [2,]  0.06683271 -0.2767573
## [3,]  0.96574373  0.1522626

(Vtilde <- X.svd$v %*% diag(sqrt(X.svd$d)) ) # coordinates of the variables

##           [,1]       [,2]
## [1,] 0.9974934  0.2408796
## [2,] 1.0041656 -0.2392791

op <- par(mfrow=c(1,2), mai=c(.8, .8, .1, .1))
nn <- paste("Indiv.",1:nrow(X))
plot(X,xlab='x1',ylab='x2',cex=1.25,cex.lab=1.25)
text(X[,1:2],nn,pos=c(4,1,2),cex=1.25)

xlim <- ylim <- range(c(Utilde, Vtilde))
plot(Utilde,xlab='First biplot component',ylab='Second biplot
     component',xlim=xlim,ylim=ylim,cex.lab=1.25)
text(Utilde[,1:2],nn,pos=c(4,1,2),cex=1.25)
arrows(x0=0, y0=0, x1=Vtilde[,1], y1=Vtilde[,2], col='red')
text(.8*Vtilde[,1], Vtilde[,2], paste('Variable',1:ncol(X)), pos=c(3,1),
     col='red', cex=1.25)
par(op)
```

Figure 34 (right) represents these coordinates in the biplot, where individuals are shown as dots and variables as arrows. Individual 3 is close to the direction of growth of variables 1 and 2, indicating that it takes a large positive value for both variables, whereas individual 1 is located in the opposite direction, indicating large negative values. Individual 2 does not have particularly large values for either variable and hence appears close to the centre of the biplot. Note that the biplot mimics what we could already see in the plot with the original observations in this simple example.

---

**Remark 4.3.3 (Relationship between biplot and principal components).**

Let $\mathtt{X} = ULV^\mathsf{T}$ be the singular value decomposition of a column-centred $n \times p$ matrix $\mathtt{X}$. This implies that we have an eigendecomposition of the form $S = V\Lambda V^\mathsf{T}$ for the sample covariance $S = \mathtt{X}^\mathsf{T}\mathtt{X}/(n-1)$.

Let the first $k$ principal components scores $\mathtt{Y}^{(k)}$ be the first $k$ columns in $\mathtt{Y} = \mathtt{X}V$, as usual. Then $Y^{(k)} = \tilde{U}_k L_k^{1/2}$, where $\tilde{U}_k$ is defined in Definition 4.3.1 and $L_k$ is the $k \times k$ upper-left submatrix of $L$. That is, the first $k$

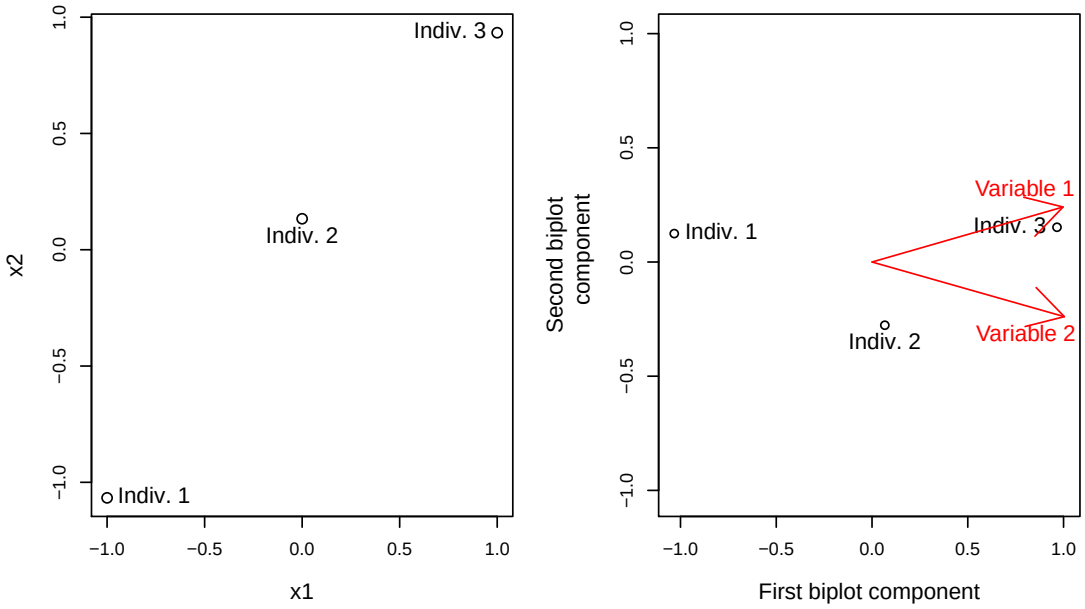



Figure 34: Biplot for data in Example 4.3.2

PC scores are scaled versions of the row coordinates from the $k$-dimensional biplot.

*Proof.* Left as an exercise. □

The implication of Definition 4.3.1 is that the coordinates of individuals (rows in `X`) in a biplot are simply a re-scaled version of their coordinates in a principal components plot.

**Example 4.3.4** (cars dataset Biplot)**.** Figure 35 shows a biplot for the cars dataset. The car positions (rows) are very similar to those in the principal component plot (Figure 23). The variables are indicated as red arrows, and help appreciate relationships amongst them and with each car. For instance, horse power (hp) and number of cylinders (cyl) grow in the same direction, whereas they are both opposed to consumption in miles per gallon (mpg) which indicades that cars with more cylinders tend to run less miles per gallon (higher consumption). The projection of cars on each arrow indicates the value of that variable for that particular car, for instance the Toyota Corolla and Fiat 128 have largest projections on mpg, suggesting their mpg is very high. The plot also shows that the cars towards the (top) right have more horse power (hp) than the one on the (bottom) left.

Here is a more in-depth interpretation of the biplot[1]. The data falls into 3 categories: 4 cylinder cars, V6 and V8. Looking at the data, you will notice the large majority of cars from America are V8, European cars was overall a mix and Japan was mainly 4 cylinder. An annotated biplot (Figure 36) shows clearly the 3 groups which are 4 cylinders cars on the left (in yellow), V6 in the middle (in purple) and V8 on the right (in green). Also the red box shows the cars which have

[1]from Alex Murphy (2018–19 cohort), which I thank for sharing it with the class.

particularly large hp. Notice that the Lincoln and the Cadillac aren't in the red square, but still have a high hp; however it is lower than those in the red box, and they also have the larger wt (weight), so you expect them to be near the wt arrow, which they are indeed.

Below is the R code used to produce Figure 35.

```
require(grDevices)
data(mtcars)
nn <- rownames(mtcars)
## print the data, ordered by decreasing hp
mtcars[mtcars$hp %>% order(decreasing=TRUE),1:7] %>% round(2)

##                      mpg cyl  disp  hp drat   wt  qsec
## Maserati Bora       15.0   8 301.0 335 3.54 3.57 14.60
## Ford Pantera L      15.8   8 351.0 264 4.22 3.17 14.50
## Duster 360          14.3   8 360.0 245 3.21 3.57 15.84
## Camaro Z28          13.3   8 350.0 245 3.73 3.84 15.41
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.34 17.42
## Lincoln Continental 10.4   8 460.0 215 3.00 5.42 17.82
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.25 17.98
## Merc 450SE          16.4   8 275.8 180 3.07 4.07 17.40
## Merc 450SL          17.3   8 275.8 180 3.07 3.73 17.60
## Merc 450SLC         15.2   8 275.8 180 3.07 3.78 18.00
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.44 17.02
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.85 17.05
## Ferrari Dino        19.7   6 145.0 175 3.62 2.77 15.50
## Dodge Challenger    15.5   8 318.0 150 2.76 3.52 16.87
## AMC Javelin         15.2   8 304.0 150 3.15 3.44 17.30
## Merc 280            19.2   6 167.6 123 3.92 3.44 18.30
## Merc 280C           17.8   6 167.6 123 3.92 3.44 18.90
## Lotus Europa        30.4   4  95.1 113 3.77 1.51 16.90
## Mazda RX4           21.0   6 160.0 110 3.90 2.62 16.46
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.88 17.02
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.21 19.44
## Volvo 142E          21.4   4 121.0 109 4.11 2.78 18.60
## Valiant             18.1   6 225.0 105 2.76 3.46 20.22
## Toyota Corona       21.5   4 120.1  97 3.70 2.46 20.01
## Merc 230            22.8   4 140.8  95 3.92 3.15 22.90
## Datsun 710          22.8   4 108.0  93 3.85 2.32 18.61
## Porsche 914-2       26.0   4 120.3  91 4.43 2.14 16.70
## Fiat 128            32.4   4  78.7  66 4.08 2.20 19.47
## Fiat X1-9           27.3   4  79.0  66 4.08 1.94 18.90
## Toyota Corolla      33.9   4  71.1  65 4.22 1.84 19.90
## Merc 240D           24.4   4 146.7  62 3.69 3.19 20.00
## Honda Civic         30.4   4  75.7  52 4.93 1.62 18.52

z <- scale(mtcars[,1:7], center=TRUE, scale=TRUE)
```

```
#a <- prcomp(z)
#plot(a$x[,1:2])
#arrows(x0=0, y0=0, a$rot[,1], a$rot[,2])

svd1 <- svd(z)
u <- svd1$u; v <- svd1$v; l <- diag(svd1$d)
#head(round(z,3))
#head(round(u %*% l %*% t(v),3))
rows <- u[,1:2] %*% sqrt(l[1:2,1:2])
cols <- v[,1:2] %*% sqrt(l[1:2,1:2])
xlim <- range(cols[,1])
ylim <- range(cols[,2])
plot(rows,xlab='First biplot component',
     ylab='Second biplot component',xlim=xlim,ylim=ylim)
text(rows[,1:2],nn,pos=3)
arrows(x0=0,y0=0,x1=cols[,1],y1=cols[,2],col='red')
text(cols[,1:2],colnames(z),pos=3,col='red')
```

---

Figure 37 shows a biplot in a concrete application. Notice that this biplot uses a scaling of the row and column coordinates that is different to the one we have introduced.

## 4.4 Canonical Correlation Analysis

A common data analysis task is to study the relationship between two (sets of) variables. When we have two univariate random variables $X$ and $Y$, a standard approach would be to plot values of $X$ against $Y$ or to compute $\mathrm{Cov}(X, Y)$. When, instead, these variables are multivariate, and we have $\mathbf{X} = (X_1, \ldots, X_p)$ and $\mathbf{Y} = (Y_1, \ldots, Y_q)$ with $p$ and $q$ not too large we may plot all pairs of variables. We may also compute the natural extension of the covariance, the cross-covariance matrix $\mathrm{Cov}(\mathbf{X}, \mathbf{Y}) = (\mathrm{Cov}(X_i, Y_j))_{ij}$ from Definition 4.1.1.

When $p$ and $q$ are large, however, it is infeasible to look at all pairwise plots. Analogous to principal component analysis, we may instead choose to look at linear projections of the variables $\mathbf{a}^T\mathbf{X}$ and $\mathbf{b}^T\mathbf{Y}$, and study the relationship between these projections. But how should we choose $\mathbf{a}$ and $\mathbf{b}$? A canonical correlation analysis chooses these vectors so as to maximise the correlation between the projections.

**Definition 4.4.1 (Canonical variables).** Let $\mathbf{X} \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^q$ be random vectors with $\mathbb{E}[|\mathbf{X}|^2 + |\mathbf{Y}|^2] < \infty$. The **first pair of canonical variables** is $(\mathbf{a}_1^T\mathbf{X}, \mathbf{b}_1^T\mathbf{Y})$, where $\mathbf{a_1}, \mathbf{b_1}$ are vectors such that

$$\frac{\mathrm{Cov}(\mathbf{a}_1^T\mathbf{X}, \mathbf{b}_1^T\mathbf{Y})}{\sqrt{\mathrm{Var}(\mathbf{a}_1^T\mathbf{X})\mathrm{Var}(\mathbf{b}_1^T\mathbf{Y})}} \geq \frac{\mathrm{Cov}(\mathbf{a}^T\mathbf{X}, \mathbf{b}^T\mathbf{Y})}{\sqrt{\mathrm{Var}(\mathbf{a}^T\mathbf{X})\mathrm{Var}(\mathbf{b}^T\mathbf{Y})}} \quad \forall \mathbf{a} \in \mathbb{R}^p, \mathbf{b} \in \mathbb{R}^q.$$

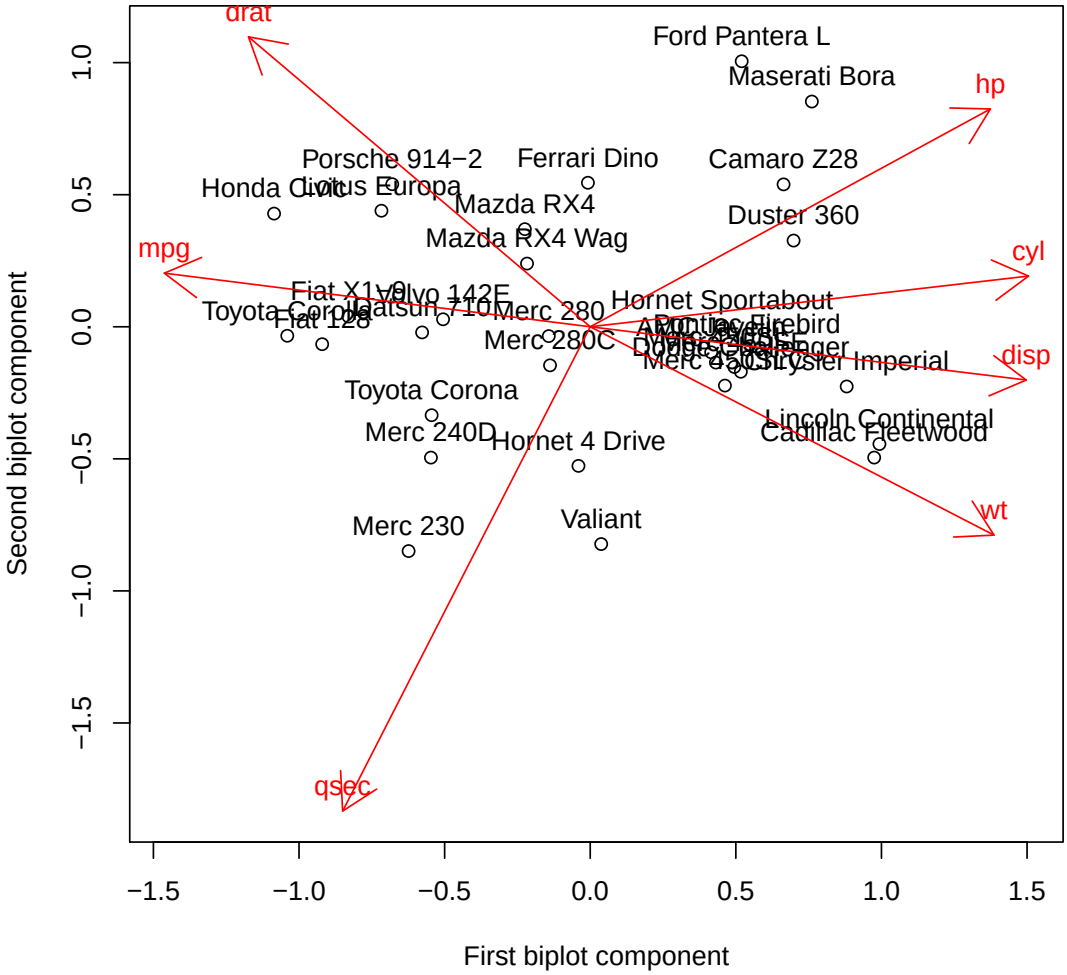



Figure 35: Biplot for the `mtcars` dataset. Points indicate cars, arrows indicate variables.

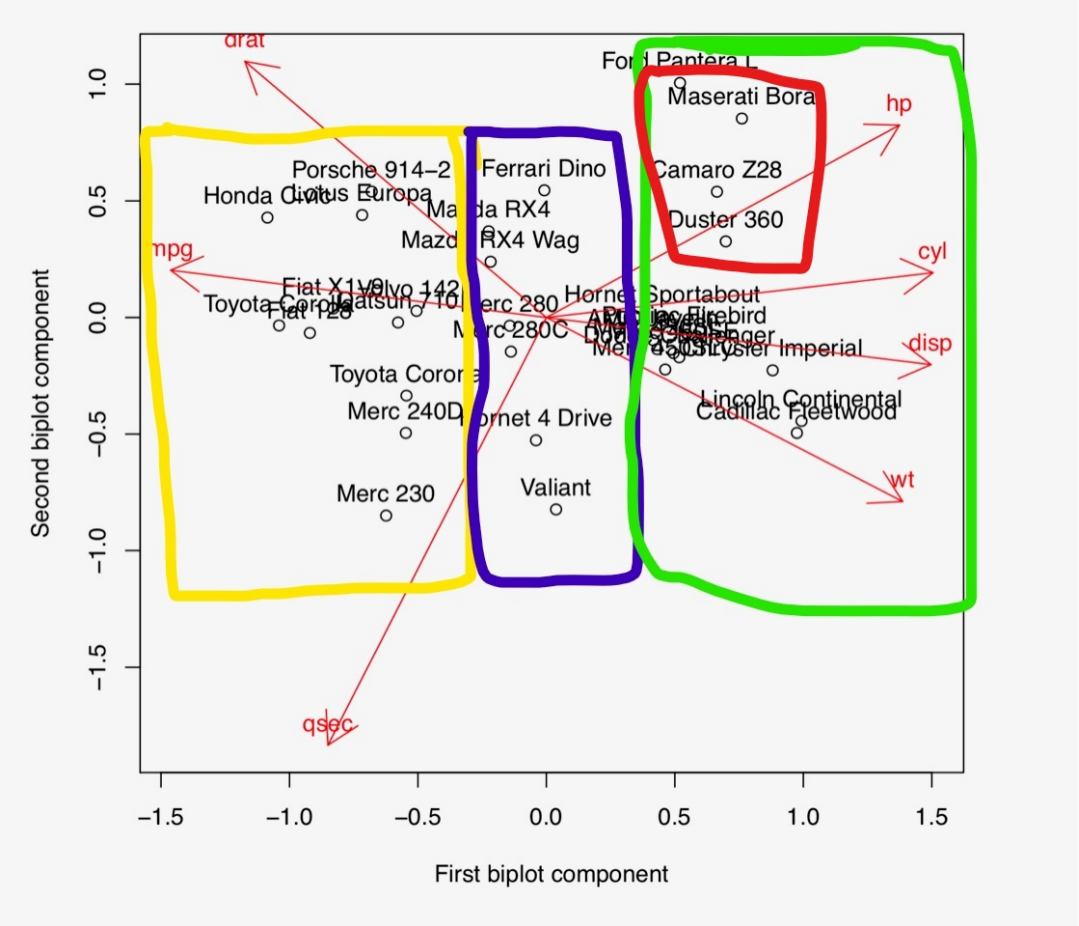



Figure 36: Annotated Biplot (by Alex Murphy).

For $k = 2, \ldots, \min(p, q)$, the **$k$-th pair of canonical variables** is $(\mathbf{a}_k^T\mathbf{X}, \mathbf{b}_k^T\mathbf{Y})$, where $\mathbf{a}_k, \mathbf{b}_k$ are chosen to maximise

$$\frac{\mathrm{Cov}(\mathbf{a}^T\mathbf{X}, \mathbf{b}^T\mathbf{Y})}{\sqrt{\mathrm{Var}(\mathbf{a}^T\mathbf{X})\mathrm{Var}(\mathbf{b}^T\mathbf{Y})}}$$

over all vectors $\mathbf{a} \in \mathbb{R}^p, \mathbf{b} \in \mathbb{R}^q$ such that

$$\mathrm{Cov}(\mathbf{a}^T\mathbf{X}, \mathbf{a}_j^T\mathbf{X}) = \mathrm{Cov}(\mathbf{b}^T\mathbf{Y}, \mathbf{b}_j^T\mathbf{Y}) = 0 \quad \text{for } j = 1, \ldots, k-1.$$

Some remarks are in order.

**Remark 4.4.2.**

1. You should compare this with the definition of principal components (Definition 4.2.1). There we maximised variances, whereas here we maximise *correlations*, which are standardised covariances.

2. As with PCA, these variables are not unique. In PCA we fixed $|\mathbf{v}_k| = 1$. In CCA it is common to make the corresponding restriction that

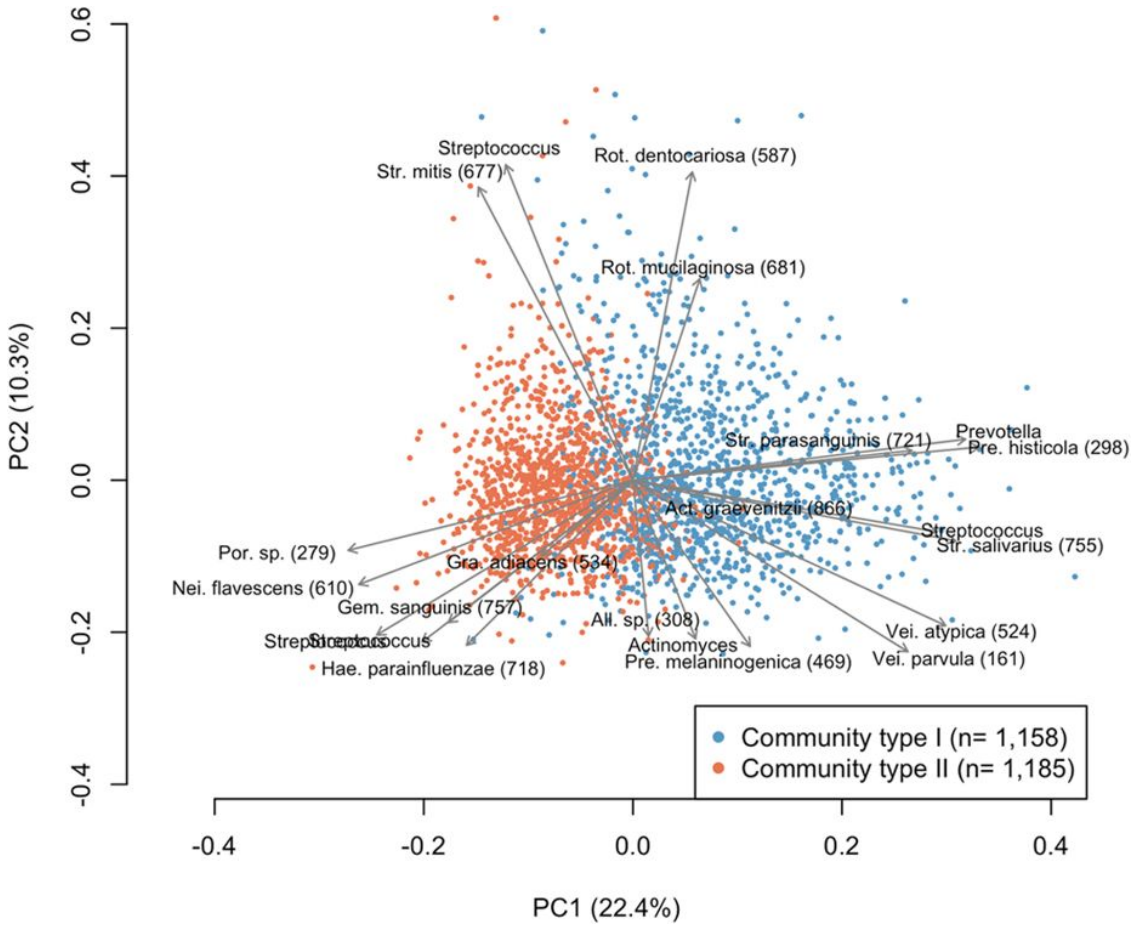



Figure 37: A biplot taken from Lim, Y., Totsika, M., Morrison, M., and Punyadeera, C. (2017). The saliva microbiome profiles are minimally affected by collection method or DNA extraction protocols. *Scientific reports*, 7(1), 8523.

$\mathrm{Var}(\mathbf{a}_k^T\mathbf{X}) = \mathrm{Var}(\mathbf{b}_k^T\mathbf{Y}) = 1$. This just changes the scale of the canonical variables.

As with PCA (cf. Theorem 4.2.3), these canonical variables can be given in a more explicit form. For simplicity we will restrict to the setting where $\mathrm{Cov}(\mathbf{X}) = I_p$ and $\mathrm{Cov}(\mathbf{Y}) = I_q$, but you may wish to try and extend the result to hold in more generality.

**Theorem 4.4.3.** Let $\mathbf{X} \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^q$ be random vectors with $\mathbb{E}[|\mathbf{X}|^2 + |\mathbf{Y}|^2] < \infty$ and $\Sigma_X = \mathrm{Cov}(\mathbf{X}) = I_p$ and $\Sigma_Y = \mathrm{Cov}(\mathbf{Y}) = I_q$. Write $\Sigma$ for the $p \times q$ cross-covariance matrix of $\mathbf{X}$ and $\mathbf{Y}$, and write

$$\Sigma = ULV^T = \ell_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \ell_r \mathbf{u}_r \mathbf{v}_r^T$$

for its singular value decomposition, where $r = \min(p, q)$. The canonical variables are given by $(\mathbf{u}_1^T\mathbf{X}, \mathbf{v}_1^T\mathbf{Y}), \ldots, (\mathbf{u}_r^T\mathbf{X}, \mathbf{v}_r^T\mathbf{Y})$.

*Proof.* See video. □

# 5 Multivariate Inference

In the previous Section, we saw descriptive methods for multivariate data. We now focus on probability theory and statistical inference. In Section 5.1 we introduce some common probability distributions for random vectors and random matrices. Section 5.2 discusses how to estimate their parameters, provide confidence intervals and confidence regions, Section 5.3 shows how to test hypotheses related to the mean or covariance of multivariate Normal distributions, and Section 5.4 shows how to check multivariate normality.

## 5.1 Multivariate probability distributions

Throughout this section we let $\mathbf{X} = (X_1, \ldots, X_p)^\mathsf{T}$ be a random vector taking values in $\mathbb{R}^p$, and let $\mathbf{x} = (x_1, \ldots, x_p)$ be an observed value of $\mathbf{X}$. Also, $\Sigma$ will denote a symmetric positive semi-definite matrix (SPD).

### 5.1.1 Multivariate Normal distribution

The multivariate normal distribution is one of the most important multivariate distributions. It is defined as follows.

**Definition 5.1.1 (Multivariate Normal Distribution).** A random vector $\mathbf{X} \in \mathbb{R}^p$ is said to follow a multivariate normal distribution (MVN) if

$$\mathbf{X} = \boldsymbol{\mu} + A\mathbf{Z},$$

for some non-random $\boldsymbol{\mu} \in \mathbb{R}^p$, some non-random $p \times l$ matrix $A$, and $\mathbf{Z} = (Z_1, \ldots, Z_l)^\mathsf{T}$, where $Z_1, \ldots, Z_l \overset{\text{iid}}{\sim} N(0,1)$, $l \geq 1$.

Notice that $\mathbb{E}\,\mathbf{X} = \boldsymbol{\mu}$ and $\mathrm{Cov}(\mathbf{X}) = AA^\mathsf{T}$, and both are well defined. Furthermore, for any $\mathbf{t} \in \mathbb{R}^p$, $\mathbf{t}^\mathsf{T}\mathbf{X} = \mathbf{t}^\mathsf{T}\boldsymbol{\mu} + (\mathbf{t}^\mathsf{T}A)\mathbf{Z}$, and therefore $\mathbf{t}^\mathsf{T}\mathbf{X}$ is a normal random variable, with mean $\mathbb{E}\left[\mathbf{t}^\mathsf{T}\mathbf{X}\right] = \mathbf{t}^\mathsf{T}\boldsymbol{\mu}$ and variance $\mathrm{Var}(\mathbf{t}^\mathsf{T}\mathbf{X}) = \mathbf{t}^\mathsf{T}AA^\mathsf{T}\mathbf{t}$. This implies the following result.

**Proposition 5.1.2 (MGF of MVN).** The moment generating function (MGF) of a MVN $\mathbf{X} = \mu + A\mathbf{Z}$ is

$$\mathbb{E}\left[\exp(\mathbf{t}^\mathsf{T}\mathbf{X})\right] = \exp\left(\mathbf{t}^\mathsf{T}\boldsymbol{\mu} + \frac{1}{2}\mathbf{t}^\mathsf{T}AA^\mathsf{T}\mathbf{t}\right), \quad \mathbf{t} \in \mathbb{R}^p.$$

Thus the distribution of $\mathbf{X}$ is uniquely defined by $\boldsymbol{\mu} = \mathbb{E}\,\mathbf{X}$ and $AA^\mathsf{T} = \mathrm{Cov}(\mathbf{X})$. We therefore write $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$, where $\Sigma = AA^\mathsf{T}$.

**Example 5.1.3.** Notice in particular that if $\mathbf{Z} = (Z_1, \ldots, Z_l)^\mathsf{T}$ is defined as above, then $\mathbf{Z} \sim N_l(0, I)$, where $I$ is the $l \times l$ identity matrix. We say that $\mathbf{Z}$ follows an **$l$-dimensional standard normal distribution**, or that it is an **$l$-dimensional standard normal random vector**.

**Example 5.1.4.** $X = 0 \cdot Y \in \mathbb{R}$, where $Y \sim N(0,1)$, is multivariate normal, with mean 0 and covariance 0. $X$ takes value 0 with probability 1. Therefore multivariate normal distributions contain as special cases distributions that are point masses.

---

**Example 5.1.5.** $\mathbf{X} = (Y_1, 0)^\mathsf{T} = A\mathbf{Y}$, where

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

is MVN. Notice that $\mathbb{P}(\mathbf{X} \in \mathbb{R} \times \{0\}) = 1$, thus $\mathbf{X}$ takes value only on the vertical line $x = 0$. This is an example of a MVN distribution which is singular distribution (it is concentrated on a set of measure 0 with respect to the Lebesgue measure on $\mathbb{R}^2$) but not a point mass.

---

Notice that by definition of the MVN distribution, we directly get the following result, which tells us that linear transformations of a multivariate normal random vector is multivariate normal.

**Proposition 5.1.6 (Affine transformation of MVN).** If $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$, $\boldsymbol{\nu} \in \mathbb{R}^q$ is non-random, and $B$ is a non-random $q \times p$ matrix, then

$$\boldsymbol{\nu} + B\mathbf{X} \sim N_q(\boldsymbol{\nu} + B\boldsymbol{\mu}, B\Sigma B^\mathsf{T}).$$

*Proof.* Left as an exercise. □

**Remark 5.1.7 (Subvectors of MVN are MVN).** This result tells us in particular that subvectors of a MVN random vector are MVN, with mean and covariance given by the corresponding sub-vectors and submatrices.

**Example 5.1.8.** Let $\mathbf{X} = (X_1, X_2)^\mathsf{T}$, where $X_1$ measures the average score that students obtain the first year of college and $X_2$ that in the second year. Suppose that $\mathbf{X} \sim N_2(\boldsymbol{\mu}, \Sigma)$ with $\boldsymbol{\mu} = (8.0, 8.2)^\mathsf{T}$ and

$$\Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}.$$

Consider $Y_1 = (X_1 + X_2)/2$, $Y_2 = X_2 - X_1$. Clearly, $\mathbf{Y} = C\mathbf{X}$ where

$$C = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 1 \end{pmatrix}.$$

By Proposition 5.1.6, the distribution of $\mathbf{Y} = (Y_1, Y_2)^\mathsf{T}$ is a bivariate Normal, with

$$\boldsymbol{\mu}_y = C\boldsymbol{\mu} = \begin{pmatrix} \frac{1}{2}(8.0 + 8.2) \\ 8.2 - 8.0 \end{pmatrix} = \begin{pmatrix} 8.1 \\ 0.2 \end{pmatrix},$$

$$\Sigma_y = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -1 & 1 \end{pmatrix} \Sigma \begin{pmatrix} \frac{1}{2} & -1 \\ \frac{1}{2} & 1 \end{pmatrix} = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.4 \end{pmatrix}.$$

Proposition 5.1.6 also tells us that $Y_1 \sim N(8.1, 0.9)$ and $Y_2 \sim N(0.2, 0.4)$, i.e. the marginal distributions of $Y_1$ and $Y_2$ are univariate Normal.

---

So far, $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$ means that $\mathbf{X}$ is an affine transformation of some $l$-dimensional standard normal random vector. The following lemma tells us that we can choose any $A$ in the definition of MVN, provided that $AA^\mathsf{T} = \Sigma$.

**Lemma 5.1.9 (MVN technical lemma).** Assume $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$. Then for any $p \times q$ matrix $B$ satisfying $\Sigma = BB^\mathsf{T}$, $\mathbf{X} = \boldsymbol{\mu} + B\widetilde{\mathbf{Z}}$, for some $\widetilde{\mathbf{Z}} \sim N_q(0, I)$.

*Proof.* Non-examinable. □

Using this Lemma, we can show that if two random vectors are jointly MVN, they are independent if and only if they are uncorrelated.

**Proposition 5.1.10 (MVN and independence).** Let $(\mathbf{X}^\mathsf{T}, \mathbf{Y}^\mathsf{T})^\mathsf{T} \sim N_{p+q}(\boldsymbol{\mu}, \Sigma)$, where $\mathbf{X} \in \mathbb{R}^p, \mathbf{Y} \in \mathbb{R}^q$, $\boldsymbol{\mu}^\mathsf{T} = (\boldsymbol{\mu}_x^\mathsf{T}, \boldsymbol{\mu}_y^\mathsf{T})$, and

$$\Sigma = \begin{pmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy}^\mathsf{T} & \Sigma_y \end{pmatrix}.$$

Then $\mathbf{X}$ and $\mathbf{Y}$ are independent if and only if $\Sigma_{xy} = 0$.

*Proof.* See video. □

We now turn to an example where $\Sigma$ is of rank 1.

**Example 5.1.11.** Let $\mathbf{X} \in N_p(0, \Sigma)$, where $\Sigma$ is of rank 1. Then $\Sigma = \lambda \mathbf{e}\mathbf{e}^\mathsf{T}$, for some $\mathbf{e} \in \mathbb{R}^p$ with $|\mathbf{e}| = 1$. Taking $A = \sqrt{\lambda}\mathbf{e}$ , we have $AA^\mathsf{T} = \Sigma$ and therefore, using Lemma 5.1.9, $\mathbf{X} = \sqrt{\lambda}\mathbf{e}Z_1$, for some $\mathbf{Z} \sim N(0, 1)$. Therefore $\mathbf{X}$ is like a $N(0, \lambda)$ random variable, but on the line $\{s\mathbf{e} : s \in \mathbb{R}\}$, and $\mathbf{X}$ has a singular distribution.

---

We can expand this previous example into the following key result, which is known as the Karhunen–Loève expansion.

**Theorem 5.1.12 (Karhunen–Loève Expansion for MVN).**
Let $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$, and let $\Sigma = \sum_{i=1}^q \lambda_i \mathbf{e}_i \mathbf{e}_i^\mathsf{T}$ be the spectral decomposition of $\Sigma$, where $q$ is the rank of $\Sigma$. Then

$$\mathbf{X} = \boldsymbol{\mu} + \sqrt{\lambda_1} Z_1 \mathbf{e}_1 + \sqrt{\lambda_2} Z_2 \mathbf{e}_2 + \cdots + \sqrt{\lambda_q} Z_q \mathbf{e}_q,$$

where $Z_1, \ldots, Z_q \overset{\text{iid}}{\sim} N(0, 1)$. In particular, $\mathbf{X}$ takes values on the affine subspace of dimension $q$ defined by

$$\{\boldsymbol{\mu} + s_1\mathbf{e}_1 + s_2\mathbf{e}_2 + \cdots + s_q\mathbf{e}_q | s_1, \ldots, s_q \in \mathbb{R}\},$$

where $\mathbf{e}_1, \ldots, \mathbf{e}_q$ are all the eigenvectors of $\Sigma$ with non-zero eigenvalues. In other words, $\mathbf{X}$ takes values only in $\boldsymbol{\mu}+\mathcal{V}$, where $\mathcal{V}$ is the subspace spanned by the eigenvectors of $\Sigma$ with non-zero eigenvalues. In particular, if $q < p$, then $\mathbf{X}$ is singular, and has no density in $\mathbb{R}^p$.

*Proof.* Left as an exercise (use the spectral decomposition of $\Sigma$). □

So now we know that not all MVN variables $\mathbf{X}$ have a density. However, if $\mathbf{X}$ has a density, then it is given by the following result.

**Proposition 5.1.13 (Density of MVN).** Let $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$. Then $\mathbf{X}$ has a density if $\Sigma$ is positive definite (or equivalently if $\Sigma$ has rank $p$, or equivalently if $\Sigma$ is invertible). In this case, its density is given by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} \det(\Sigma)^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{5.1.1}$$

*Proof.* If $\Sigma$ is has rank $p$, then $\Sigma = AA^\mathsf{T}$ for some invertible $p \times p$ matrix $A$, and $\mathbf{X} = \boldsymbol{\mu} + A\mathbf{Z}$, where $\mathbf{Z} \sim N_p(0, I)$. By independence, the density of $\mathbf{Z}$ is given by

$$f_\mathbf{Z}(z_1, \ldots, z_p) = \prod_{i=1}^{p} \left((2\pi)^{-1/2} \exp(-\frac{1}{2}z_i^2)\right) = (2\pi)^{-p/2} \exp(-\mathbf{z}^\mathsf{T}\mathbf{z}/2),$$

Now defining the transformation $\phi : \mathbb{R}^p \to \mathbb{R}^p$ by $\phi(\mathbf{z}) = \boldsymbol{\mu} + A\mathbf{z}$, we have $\phi^{-1}(\mathbf{x}) = A^{-1}(\mathbf{x}-\boldsymbol{\mu})$,

$$J_{\phi^{-1}} = \det\left(\frac{d\phi^{-1}}{d\mathbf{x}}(\mathbf{x})\right) = \det\left(A^{-1}\right) = \det\left(\Sigma\right)^{-1/2},$$

since $\det(\Sigma) = \det(AA^\mathsf{T}) = \det(A)^2$. Using the formula for the change of variables, we get

$$f_\mathbf{X}(\mathbf{x}) = f_\mathbf{Z}(\phi^{-1}(\mathbf{x}))\,|J_{\phi^{-1}}| = (2\pi)^{-p/2} \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

since

$$\begin{aligned}(\phi^{-1}(\mathbf{x}))^\mathsf{T}(\phi^{-1}(\mathbf{x})) &= (A^{-1}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}(A^{-1}(\mathbf{x}-\boldsymbol{\mu})) \\ &= (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}(A^{-1})^\mathsf{T}A^{-1}(\mathbf{x}-\boldsymbol{\mu}) \\ &= (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}).\end{aligned}$$

□

Notice that $\Sigma^{-1}$ appears in the density function, and it therefore makes sense that $\Sigma$ needs to be invertible for a density to exist, and this is equivalent to assuming $\Sigma$ is of full rank, or equivalently positive definite.

**Example 5.1.14.** The univariate Normal distribution is the case $p = 1$. To see this, set $p = 1$ in (5.1.1) and obtain

$$f(x_1) = \frac{1}{\sqrt{2\pi\sigma_{11}}} \exp\left\{-\frac{1}{2}\frac{(x_1-\mu_1)^2}{\sigma_{11}}\right\},$$

which is the density function for a univariate Normal variable. Notice here that $\sigma_{11} = \mathrm{Var}(X_1)$.

---

**Example 5.1.15.** Let us write down the density for the bivariate ($p = 2$) Normal distribution in an expanded form. The covariance is

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{pmatrix}$$

and hence $\det(\Sigma) = \sigma_{11}\sigma_{22} - \sigma_{12}^2 = \sigma_{11}\sigma_{22}(1-\rho_{12}^2)$, where $\rho_{12}$ is the correlation. The inverse of the covariance is

$$\Sigma^{-1} = \frac{1}{\sigma_{11}\sigma_{22}(1-\rho_{12}^2)} \begin{pmatrix} \sigma_{22} & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11} \end{pmatrix},$$

and hence

$$\begin{aligned}
&(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \\
&\quad = \frac{\sigma_{22}(x_1-\mu_1)^2 + \sigma_{11}(x_2-\mu_2)^2 - 2\sigma_{12}(x_1-\mu_1)(x_2-\mu_2)}{\sigma_{11}\sigma_{22}(1-\rho_{12}^2)} \\
&\quad = \frac{1}{1-\rho_{12}^2}\left(\frac{(x_1-\mu_1)^2}{\sigma_{11}} + \frac{(x_2-\mu_2)^2}{\sigma_{22}} - 2\rho_{12}\frac{x_1-\mu_1}{\sqrt{\sigma_{11}}}\frac{x_2-\mu_2}{\sqrt{\sigma_{22}}}\right) \\
&\quad = \frac{1}{1-\rho_{12}^2}\left(z_1^2 + z_2^2 - 2\rho_{12}z_1z_2\right),
\end{aligned}$$

where $z_i = (x_i-\mu_i)/\sqrt{\sigma_{ii}}$ is the z-score for $x_i$. The density is

$$f(\mathbf{x}) = \frac{1}{2\pi\sqrt{\sigma_{11}\sigma_{22}(1-\rho_{12}^2)}} \exp\left\{-\frac{1}{2(1-\rho_{12}^2)}\left(z_1^2 + z_2^2 - 2\rho_{12}z_1z_2\right)\right\}.$$

---

Notice that if $\Sigma = (\sigma_{ij})$ is the covariance matrix of $\mathbf{X}$, then the variances of the coordinates of $\mathbf{X}$ are $\sigma_{ii}$, and not $\sigma_{ii}^2$. This is different to the usual notation that you are used to for the univariate normal distribution.

Looking back at Proposition 5.1.13, we notice that the density of a MVN depends on $\mathbf{x}$ only through

$$(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}). \tag{5.1.2}$$

The quantity (5.1.2) is the square of the so-called **Mahalanobis distance between $\mathbf{x}$ and $\boldsymbol{\mu}$**, defined as $\sqrt{(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}$. We shall now see that the Mahalanobis distance is intimately related to ellipses/ellipsoids in $\mathbb{R}^p$. We start by recalling the definition of an ellipsoid.

**Definition 5.1.16 (Ellipsoid).** An ellipsoid with center $\mathbf{m} \in \mathbb{R}^p$ and axes along the coordinate axes is the set of all points $\mathbf{z} \in \mathbb{R}^p$ such that

$$\sum_{i=1}^{p} a_i(z_i - m_i)^2 = 1$$

for some $a_1, \ldots, a_p \in \mathbb{R}^+$. Further, the length of the $i^{th}$ axis of the ellipsoid is $a_i^{-\frac{1}{2}}$.

**Proposition 5.1.17 (Contours of MVN density).** The contours of constant density $(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = c$ for a $N_p(\boldsymbol{\mu}, \Sigma)$ distribution define an ellipsoid centered at $\boldsymbol{\mu}$ whose axes are given by $\mathbf{e}_1, \ldots, \mathbf{e}_p$, the eigenvectors of $\Sigma$. Further, the length of axis $i$ of the ellipsoid is given by $\sqrt{c\lambda_i}$, where $\lambda_1, \ldots, \lambda_p$ are the eigenvalues of $\Sigma$.

This result gives a natural interpretation of Principal Component Analysis when the data follow a multivariate Normal. The Principal Components are obtained by projecting $\mathbf{X}$ on the ellipsoid axes.

*Proof.* We start by noting that the eigenvectors of $\Sigma^{-1}$ are the same as those of $\Sigma$, and its eigenvalues are the inverse $\lambda_1^{-1}, \ldots, \lambda_p^{-1}$. Therefore, using the eigendecomposition of $\Sigma^{-1}$ we can write

$$(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = (\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\left(\sum_{i=1}^{p}\frac{1}{\lambda_i}\mathbf{e}_i\mathbf{e}_i^\mathsf{T}\right)(\mathbf{x}-\boldsymbol{\mu}) =$$
$$\sum_{i=1}^{p}\frac{1}{\lambda_i}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{e}_i\mathbf{e}_i^\mathsf{T}(\mathbf{x}-\boldsymbol{\mu}) = \sum_{i=1}^{p}\frac{1}{\lambda_i}(z_i - m_i)^2, \tag{5.1.3}$$

where $z_i = \mathbf{x}^\mathsf{T}\mathbf{e}_i$ and $m_i = \boldsymbol{\mu}^\mathsf{T}\mathbf{e}_i$ are the projections on $\mathbf{e}_i$, which therefore define the axes of the ellipsoid. Setting (5.1.3) equal to $c$ gives

$$\sum_{i=1}^{p}\frac{1}{c\lambda_i}(z_i - m_i)^2 = 1,$$

which is an ellipsoid with axis lengths $\sqrt{c\lambda_i}$. □

```
library(mvtnorm)
library(magrittr)

xseq <- seq(-4,4,length=100)
xgrid <- expand.grid(xseq,xseq)
mu <- c(0,0); S <- matrix(c(2,1,1,1),nrow=2)
y <- dmvnorm(xgrid, mean=mu, sigma=S)
```

```
zlim <- range(y);
e <- eigen(S)$vectors

par(mfrow=c(2,1), mar=c(0,0,0,0))
zlim[1] <- -.1
ctrs <- contourLines(x= xseq, y=xseq,
                     z=matrix(y,nrow=length(xseq),ncol=length(xseq)),
                     nlevels=10)
ctrs[[1]] %>% str

## List of 3
##  $ level: num 0.02
##  $ x    : num [1:245] -2.87 -2.88 -2.88 -2.87 -2.87 ...
##  $ y    : num [1:245] -1.56 -1.49 -1.41 -1.33 -1.31 ...

V <- persp(x=xseq, y=xseq, z=matrix(y,nrow=length(xseq),
        ncol=length(xseq)), xlab='X1', ylab='X2',
        zlab='Density', theta=10, phi=30, border=NA,
        col=grey(.9), box=TRUE , shade=.75, zlim=zlim)
for(i in 1:length(ctrs)) ## plot the contours
    lines(trans3d(x=ctrs[[i]]$x, y=ctrs[[i]]$y, z=zlim[1], V), col=1)
points(trans3d(x=mu[1], y=mu[2], z=zlim[1], V), col=1, pch=20)
lines(trans3d(x=c(-10,10)*e[1,1], y=c(-10,10)*e[2,1], z=zlim[1], V),
      col=2)
lines(trans3d(x=c(-10,10)*e[1,2], y=c(-10,10)*e[2,2], z=zlim[1], V),
      col=4, lty=2)
par(mai=c(.8,.8,.8,.8))
contour(x=xseq, y=xseq, z=matrix(y,nrow=length(xseq),ncol=length(xseq)),
        xlab='X1', ylab='X2', asp=1, xlim=3*c(-1,1), ylim=3*c(-1,1))
points(mu[1],mu[2],pch=20,cex=1)
#
segments(x0=-100*e[1,1],y0=-100*e[2,1], x1=100*e[1,1],y1=100*e[2,1],
         col=2)
segments(x0=-100*e[1,2],y0=-100*e[2,2], x1=100*e[1,2],y1=100*e[2,2],
         col=4, lty=2)
```

**Example 5.1.18.** Consider a bivariate Normal distribution with $\boldsymbol{\mu} = (0,0)^\mathsf{T}$ and $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$. A plot of the density and its contours are given in Figure 38.

We observe that the contours are elliptical. The eigenvectors of $\Sigma$ are $\mathbf{e}_1 = (0.851, 0.526)^\mathsf{T}$ and $\mathbf{e}_2 = (0.526, -0,851)^\mathsf{T}$, with corresponding eigenvalues $\lambda_1 = 2.62$ and $\lambda_2 = 0.38$. The lower panel in Figure 38 shows the axis defined by $\mathbf{e}_1$ as a solid line and that for $\mathbf{e}_2$ as a dashed line. The length of the first and second axes are $\sqrt{2.62} = 1.62$ and $\sqrt{0.38} = 0.62$, respectively. This matches our visual impression that the main axis is 2-3 longer than the secondary axis.

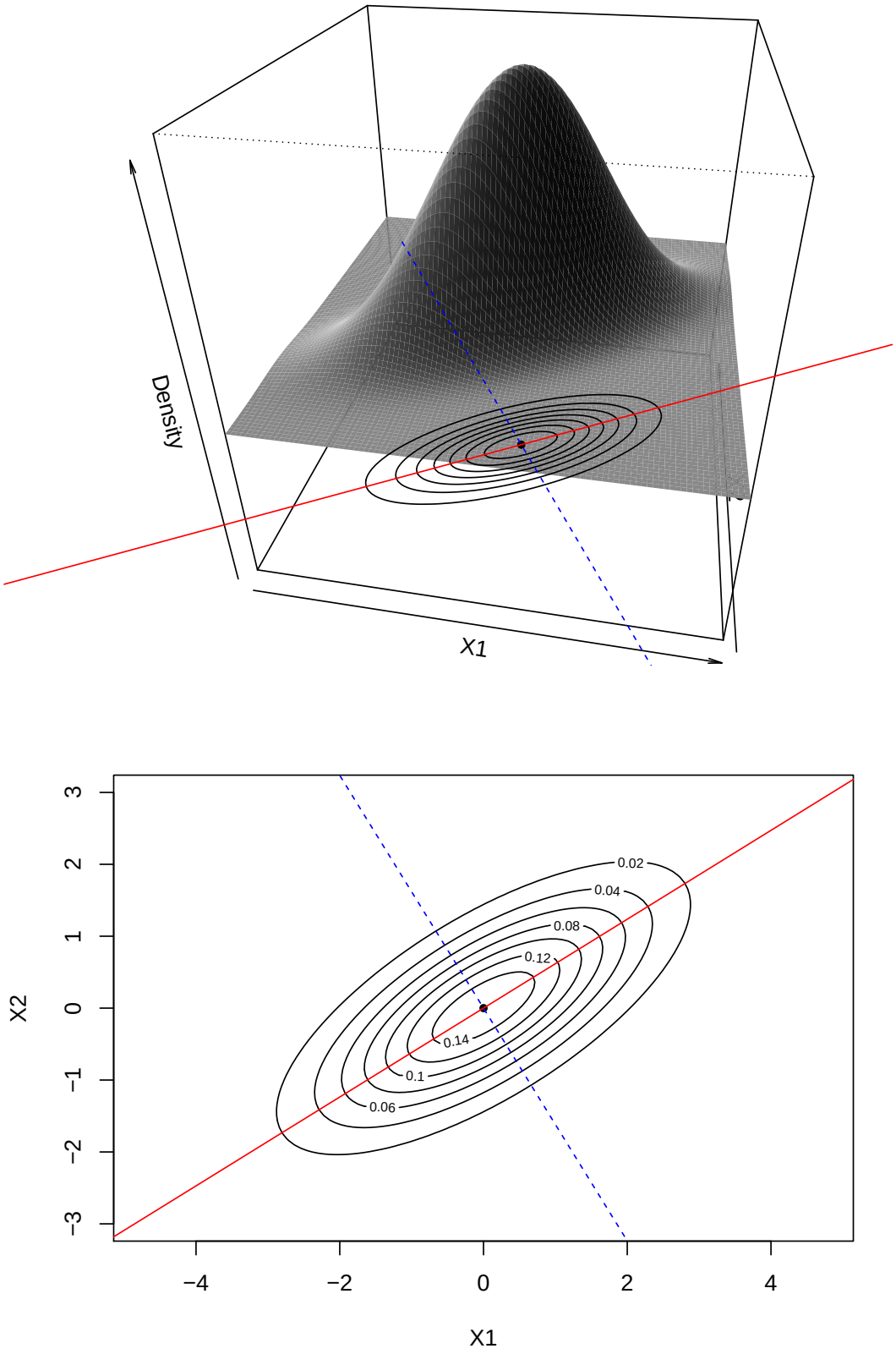



Figure 38: Bivariate Normal with $\boldsymbol{\mu} = (0, 0)$, $\sigma_{11} = 2, \sigma_{22} = 1, \sigma_{12} = 1$. Top: density function, with contours; Bottom: density contours. In both plots, the first eigenvector is in red, and the second eigenvector is in dashed blue.

So far we have seen that the multivariate Normal has the nice properties that its margins are again multivariate Normal and that it is closed with respect to linear combinations. Another interesting property is that the conditional distributions are also multivariate Normal.

**Proposition 5.1.19 (Conditional distributions of MVN).**
Let $\mathbf{X}^\mathsf{T} = (\mathbf{X}_1^\mathsf{T}, \mathbf{X}_2^\mathsf{T}) \sim N_p(\boldsymbol{\mu}, \Sigma)$ with $\boldsymbol{\mu}^\mathsf{T} = (\boldsymbol{\mu}_1^\mathsf{T}, \boldsymbol{\mu}_2^\mathsf{T})$ and

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^\mathsf{T} & \Sigma_{22} \end{pmatrix},$$

where $\mathbf{X}_1$ is $q \times 1$, $\mathbf{X}_2$ is $(p-q) \times 1$, $\Sigma_{11}$ is $q \times q$ and $\Sigma_{22}$ is $(p-q) \times (p-q)$. Further, $\Sigma_{22}$ is assumed to be of full rank.

Then the conditional distribution of $\mathbf{X}_1$ given $\mathbf{X}_2$ is

$$\mathbf{X}_1 \mid \mathbf{X}_2 \sim N_q(\boldsymbol{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{X}_2 - \boldsymbol{\mu}_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^\mathsf{T}).$$

Notice in particular that this result implies that $A\mathbf{X}|B\mathbf{X}$ is multivariate normal if $B\mathbf{X}$ has a density.

*Proof.* Left as an exercise. Hint: use Proposition 5.1.10 to show that

$$\mathbf{X}_1 - \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{X}_2 - \boldsymbol{\mu}_2) \quad \text{and} \quad \mathbf{X}_2$$

are independent. □

**Example 5.1.20.** Let $\mathbf{X} = (X_1, X_2)^\mathsf{T}$ follow a bivariate Normal with mean $\boldsymbol{\mu} = (\mu_1, \mu_2)^\mathsf{T}$ and covariance

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{pmatrix}.$$

Following Proposition 5.1.19, the distribution of $X_1$ given $X_2$ is a univariate Normal with variance

$$\mathrm{Var}(X_1 \mid X_2) = \sigma_{11} - \frac{\sigma_{12}^2}{\sigma_{22}} = \sigma_{11}(1 - \rho_{12}^2)$$

and mean

$$\mathbb{E}(X_1 \mid X_2) = \mu_1 + \frac{\sigma_{12}}{\sigma_{22}}(X_2 - \mu_2).$$

As one would expect, $\mathrm{Var}(X_1 \mid X_2) = \sigma_{11} = \mathrm{Var}(X_1)$ when $\rho_{12} = 0$ and $\mathrm{Var}(X_1 \mid X_2) \to 0$ when $\rho \to \pm 1$. Also, if $\mathrm{Cov}(X_1, X_2) = \sigma_{12} > 0$ we see that $\mathbb{E}(X_1 \mid X_2)$ grows as $(X_2 - \mu_2)$ increases. When $\sigma_{12} < 0$ then $\mathbb{E}(X_1 \mid X_2)$ decreases as $(X_2 - \mu_2)$ increases.

---

Based on all we have seen so far it is straightforward to see that one can transform $\mathbf{X}$ into a set of independent variables, and that squared Mahalanobis distance between $\mathbf{X}$ and $\boldsymbol{\mu}$ follows a chi-square distribution with $p$ degrees of freedom. The latter result can be useful to identify outliers, as it tells us the region that should contain observations from $\mathbf{X}$ with probability $1 - \alpha$. Before stating the next result, let

us recall the definition of a chi-square random variable: $Z \sim \chi^2_p$ if $Z = Y_1^2 + \cdots + Y_p^2$ for some $Y_1, \ldots, Y_p \overset{\text{iid}}{\sim} N(0,1)$. That is, the $\chi^2_p$ distribution is the distribution of the squared norm of a $N_p(0, I)$ random vector.

**Proposition 5.1.21 (Transformation to independent variables).** Let $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$. If $\Sigma$ has full rank, then

$$\mathbf{Y} = \Sigma^{-1/2}(\mathbf{X} - \boldsymbol{\mu}) \sim N_p(\mathbf{0}, I_p).$$

Furthermore,

$$D(\mathbf{X}) \triangleq (\mathbf{X} - \boldsymbol{\mu})^\mathsf{T} \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu}) \sim \chi^2_p,$$

a chi-square distribution with $p$ degrees of freedom. Thus

$$\mathbb{P}\left((\mathbf{X} - \boldsymbol{\mu})^\mathsf{T} \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu}) \leq \chi^2_p(1-\alpha)\right) = 1 - \alpha,$$

where $\chi^2_p(1-\alpha)$ is the 1-$\alpha$ quantile of a $\chi^2_p$ distribution.

*Proof.* Left as an exercise. □

**Example 5.1.22.** Consider the bivariate Normal from Example 5.1.18, with $\boldsymbol{\mu} = (0,0)^\mathsf{T}$ and

$$\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}.$$

We can use the R package `mvtnorm` to obtain 200 draws from this distribution. The Mahalanobis distance between each draw and $\boldsymbol{\mu}$ follows a $\chi^2_2$ distribution, which has 95% percentile equal to 5.99. Therefore, the ellipse $(\mathbf{X} - \boldsymbol{\mu})^\mathsf{T} \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu}) = 5.99$ should contain roughly 95% of the observations. The upper panel in Figure 39 shows the observed values and the ellipse (drawn with R package `ellipse`, see below). We see that indeed most of the observations lie within the ellipse (97%, to be exact).

We then compute squared Mahalanobis distances (R function `mahalanobis`) and display a histogram in the lower panel of Figure 39. We see that the histogram closely resembles the the theoretical $\chi^2_2$ distribution (gray line), and that most distances are smaller than the 95% cutoff at 5.99. To be precise, 97% of the distances are below 5.99, as we just saw. The R code required to produce these plots is provided below.

```
library(mvtnorm)
library(ellipse)


#Obtain draws
mu <- c(0,0); S <- matrix(c(2,1,1,1),nrow=2)
set.seed(1) # for reproducibility
x <- rmvnorm(n=200, mean=mu, sigma=S)

# Find 95\% quantile
```

```r
(c <- qchisq(0.95, df=2))

## [1] 5.991465

#Compute squared Mahalanobis dist.
d <- mahalanobis(x, center=mu, cov=S)
mean(d>c)

## [1] 0.03

#Plot 1
par(mfrow=c(2,1))
plot(x,xlab='X1',ylab='X2')
lines(ellipse(S, centre=mu, level=0.95))

#Plot 2
hist(d,xlab='squared Mahalanobis distance',main='',cex.lab=1.5)
abline(v=c,lwd=2,lty=2)
dseq <- seq(0,1.5*max(d),length=1000)
lines(dseq, 200*dchisq(dseq, df=2), col='gray', lwd=2)
legend('topright',c('Observed freq.','Chi-square df=2'),
       lty=1, lwd=2, col=c('black','gray'), cex=1.3)
par(op)
```

---

### 5.1.2 Wishart distribution

So far we described probability distributions for random vectors. We now turn to an important family of distributions for random matrices (matrices whose entries are random variables). The following distribution on random matrices is motivated by the fact that the sample covariance matrix if of the form $n^{-1}\sum_{i=1}^{n}\mathbf{X}_i\mathbf{X}_i^\mathsf{T}$, if $\mathbb{E}\,\mathbf{X}_i = 0$.

**Definition 5.1.23 (Wishart Distribution).**
Suppose $\mathbf{X}_1, \ldots, \mathbf{X}_n \overset{\text{iid}}{\sim} N_p(0, \Sigma)$. Then the random $p \times p$ matrix

$$S = \sum_{i=1}^{n} \mathbf{X}_i\mathbf{X}_i^\mathsf{T}$$

follows a $p$-dimensional Wishart distribution with $n$ degrees of freedom and scale matrix $\Sigma$. Since $S$ depends on $p, \Sigma, n$, we write

$$S \sim \mathcal{W}_p(\Sigma, n).$$

If $\Sigma$ is of full rank (and hence invertible), and $n \geq p$, then the density

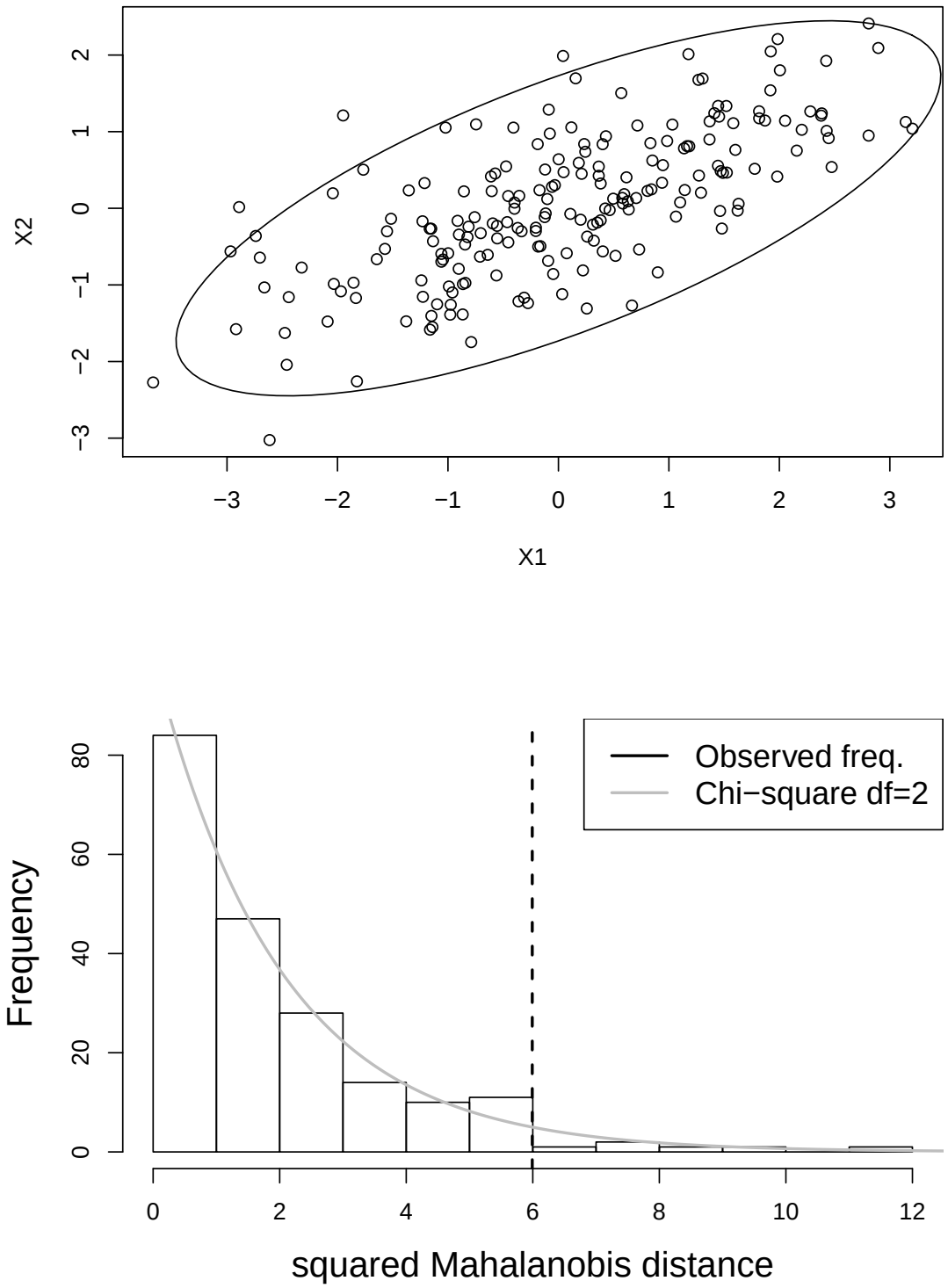



Figure 39: 200 bivariate Normal draws with mean $(0,0)^\mathsf{T}$, $\sigma_{11} = 2, \sigma_{12} = \sigma_{22} = 1$. Top: observed draws and 95% probability ellipse; Bottom: histogram with squared Mahalanobis distances (black), theoretical $\chi^2_2$ distribution (grey) and 95% theoretical quantile (black dashed line).

of a $\mathcal{W}_p(\Sigma, n)$ random matrix is given by

$$f(S) = \frac{1}{2^{\frac{np}{2}} \det(\Sigma)^{\frac{n}{2}} \Gamma_p(\frac{n}{2})} \det(S)^{\frac{n-p-1}{2}} \exp\left\{-\frac{1}{2} \operatorname{Tr}\left(\Sigma^{-1} S\right)\right\}, \quad (5.1.4)$$

for $S$ SPD, where $\Gamma_p(\cdot)$ is the multivariate gamma function. The density is zero for all $S$ that are not SPD.

You do not need to remember the formula for the density.

We note that the construction ensures that $S$ is symmetric (as each term is symmetric) and positive semi-definite. To see the latter let $\mathbf{z} \in \mathbb{R}^p$ be an arbitrary non-zero vector, then

$$\mathbf{z}^\mathsf{T} S \mathbf{z} = \mathbf{z}^\mathsf{T} \left( \sum_{i=1}^n \mathbf{X}_i \mathbf{X}_i^\mathsf{T} \right) \mathbf{z} = \sum_{i=1}^n (\mathbf{z}^\mathsf{T} \mathbf{X}_i)(\mathbf{X}_i^\mathsf{T} \mathbf{z}) = \sum_{i=1}^n (\mathbf{z}^\mathsf{T} \mathbf{X}_i)^2 \geq 0.$$

Let us see that if $\Sigma$ is invertible and $n \geq p$, then $S$ is strictly positive-definite. Note that $S = \mathtt{X}^\mathsf{T}\mathtt{X}$ where $\mathtt{X}$ is an $n \times p$ matrix where the $i^{th}$ row is equal to $\mathbf{X}_i^\mathsf{T}$. Thus $S$ has full rank if and only if at least $p$ vectors amongst $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are linearly independent, which given that $n \geq p$ happens with probability 1 under any continuous probability distribution with a density, and in particular the MVN with invertible $\Sigma$. Hence $S^{-1}$ exists which implies that none of its eigenvalues can be 0 and thus it is strictly positive definite. As a remark, note that if one were to consider the case $n < p$ then $X$ has rank $n$ and thus $S$ would not be of full rank.

We note that if $S^{-1}$ exists, then it is also positive definite and symmetric, hence this can be another interesting way to define random matrices (in fact, it gives rise to the *inverse Wishart distribution*, which we will not discuss here).

**Proposition 5.1.24 (Properties of the Wishart Distribution).**

1. The mean is $E(S) = n\Sigma$.
2. The variance of element $(i, j)$ is $\operatorname{Var}(s_{ij}) = n(\sigma_{ij}^2 + \sigma_{ii}\sigma_{jj})$ for $n \geq p+1$.
3. Additivity of degrees of freedom. If $S_1 \sim \mathcal{W}_p(\Sigma, n)$ and $S_2 \sim \mathcal{W}_p(\Sigma, m)$ are independent, then $S_1 + S_2 \sim \mathcal{W}_p(\Sigma, n+m)$.
4. If $S \sim \mathcal{W}_p(\Sigma, n)$ and $C$ is a $q \times p$ matrix, then $CSC^\mathsf{T} \sim \mathcal{W}_q(C\Sigma C^\mathsf{T}, n)$.
5. If $\mathbf{v} \in \mathbb{R}^p$, then
$$\mathbf{v}^\mathsf{T} S \mathbf{v} \sim (\mathbf{v}^\mathsf{T} \Sigma \mathbf{v}) \chi_n^2.$$
The result implies in particular that $W_1(\alpha, n) \overset{d}{=} \alpha\chi_n^2$, and that
$$(S)_{ii}/\sigma_{ii} \sim \chi_n^2$$
for $i = 1, \ldots, p$.

*Proof.* Part 1. follows from direct calculations (do it!).

Part 2. will not be shown, and is not examinable.

Part 3. follows immediately by noting that $S_1 + S_2 = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i + \sum_{j=1}^{m} \mathbf{y}_j \mathbf{y}_j^\mathsf{T}$ with independence across $i, j$, which is precisely the definition of a $\text{Wishart}_{n+m}(\Sigma)$.

For part 4. we note that by definition

$$CSC^\mathsf{T} = \sum_{i=1}^{n} C\mathbf{x}_i \mathbf{x}_i^\mathsf{T} C^\mathsf{T} = \sum_{i=1}^{n} \mathbf{z}_i \mathbf{z}_i^\mathsf{T},$$

where $\mathbf{z}_i = C\mathbf{x}_i \sim N_q(\mathbf{0}, C\Sigma C^\mathsf{T})$ are independent. Hence again by definition $CSC^\mathsf{T} \sim \mathcal{W}_q(C\Sigma C^\mathsf{T}, n)$.

Part 5. is left as an exercise. □

### 5.1.3 Hotelling's $T^2$ distribution

Recall that a Student $t$ distribution with $q$ degrees of freedom is defined as the ratio

$$t_q \stackrel{d}{=} \frac{N(0,1)}{\sqrt{\chi_q^2/q}},$$

where the variables in the ratio are independent, and $\stackrel{d}{=}$ means equal in distribution. The following distribution, called Hotelling's $T^2$ distribution, is a univariate distribution that generalizes the Student $t$ distribution, and will be important later on.

**Definition 5.1.25 (Hotelling's $T^2$ Distribution).**
A random variable $D$ follows Hotelling's $T^2$ distribution with degrees of freedom $(p, m)$, written $T^2_{p,m}$, where $m \geq p$, if and only if

$$D = \mathbf{X}^\mathsf{T} \left(\frac{S}{m}\right)^{-1} \mathbf{X},$$

for some $\mathbf{X} \sim N_p(\mathbf{0}, \Sigma)$, $S \sim \mathcal{W}_p(\Sigma, m)$, where $\Sigma$ is invertible, and $\mathbf{X}, S$ are independent.

As an exercise, show that the $T^2_{p,m}$ distribution does not depend on the choice of $\Sigma$. If we take $p = 1$, Hotelling's $T^2$ distribution is equal to the square of a student $t_m$ distribution. What if $p > 1$? Well, then it is linked to the $F$ distribution, which we now recall:

**Definition 5.1.26 (F distribution).**
Let $p, q \in \{1, 2, \ldots\}$. A random variable $W \in \mathbb{R}$ is said to follow a $F_{p,q}$ distribution, also called *$F$ distribution with $(p, q)$ degrees of freedom*, if

$$W = \frac{U/p}{V/q},$$

for some independent $U \sim \chi_p^2, V \sim \chi_q^2$.

Notice in particular that $(t_q)^2 \stackrel{d}{=} F_{1,q}$, therefore the F distribution generalizes student's $t$ distribution. It turns out that a re-scaled version of a Hotelling's $T^2$ follows an F distribution. Specifically, we have the following result.

**Proposition 5.1.27 (Link between Hotelling's and the F distribution).**
If $D \sim T^2_{p,m}$, then

$$\frac{m-p+1}{mp} D \sim F_{p,m-p+1}.$$

Hence, we can easily evaluate the density and tail probabilities using those from the F distribution. The proof of this Proposition is beyond the scope of this module.

## 5.2 Parameter estimation

We now discuss how to obtain maximum likelihood estimates for the mean and covariance of a multivariate Normal distribution, and their associated confidence regions. We will only consider the case where $\Sigma$ is invertible. If it is not, then one needs to work in coordinates with respect to the eigenvectors of $\Sigma$ that have non-zero eigenvalues (and this is beyond the scope of the module).

### 5.2.1 Point estimates

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be i.i.d. draws from a $N_p(\boldsymbol{\mu}, \Sigma)$. The likelihood function is

$$\begin{aligned} L(\boldsymbol{\mu}, \Sigma) &= \prod_{i=1}^{n} \left( \frac{1}{(2\pi)^{\frac{p}{2}} \det(\Sigma)^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})^{\mathsf{T}} \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) \right\} \right) \\ &= \frac{1}{(2\pi)^{\frac{np}{2}} \det(\Sigma)^{\frac{n}{2}}} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})^{\mathsf{T}} \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) \right\} \end{aligned} \tag{5.2.1}$$

The maximum likelihood estimate (MLE) is obtained by maximizing (5.2.1) with respect to $(\boldsymbol{\mu}, \Sigma)$. In order to find the MLE we will need the following result.

**Lemma 5.2.1 (MLE technical lemma).** Let $B$ be a symmetric positive definite (SPD) $p \times p$ matrix and $b > 0$. Then the maximum of the function

$$g(\Sigma) \triangleq \det(\Sigma)^{-b} \exp\left( -\frac{1}{2} \operatorname{Tr}(\Sigma^{-1} B) \right),$$

defined over the space of SPD matrices, is obtained for $\Sigma = \frac{1}{2b} B$.

*Proof.* Proof may be on Assignment 2. □

**Proposition 5.2.2 (MLE for the mean and covariance of MVN distribution).**

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be an observed random sample from $N_p(\boldsymbol{\mu}, \Sigma)$.

1. The MLE for $\boldsymbol{\mu}$ is
$$\hat{\boldsymbol{\mu}} \triangleq \bar{\mathbf{x}} \triangleq \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$
2. The MLE for $\Sigma$ is
$$\hat{\Sigma} \triangleq \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}$$

*Proof.* We start by maximizing (5.2.1) with respect to $\boldsymbol{\mu}$, which is equivalent to minimizing
$$\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}).$$
The optimum could be easily found using vector derivatives, but here we shall use a more basic approach. Using that the trace of a number is that same number and other properties of the trace, we re-write the expression as

$$\begin{aligned}
&\operatorname{Tr}\left(\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu})\right) = \sum_{i=1}^{n}\operatorname{Tr}\left((\mathbf{x}_i - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu})\right)\\
&= \sum_{i=1}^{n}\operatorname{Tr}\left(\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\mathsf{T}\right) = \operatorname{Tr}\left(\Sigma^{-1}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\mathsf{T}\right)\\
&= \operatorname{Tr}\left(\Sigma^{-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}} + \bar{\mathbf{x}} - \boldsymbol{\mu})(\mathbf{x}_i - \bar{\mathbf{x}} + \bar{\mathbf{x}} - \boldsymbol{\mu})^\mathsf{T}\right)\\
&= \operatorname{Tr}\left(\Sigma^{-1}\sum_{i=1}^{n}\big((\mathbf{x}_i - \bar{\mathbf{x}}) + (\bar{\mathbf{x}} - \boldsymbol{\mu})\big)\big((\mathbf{x}_i - \bar{\mathbf{x}}) + (\bar{\mathbf{x}} - \boldsymbol{\mu})\big)^\mathsf{T}\right)\\
&= \operatorname{Tr}\left(\Sigma^{-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T} + \Sigma^{-1}\sum_{i=1}^{n}(\bar{\mathbf{x}} - \boldsymbol{\mu})(\bar{\mathbf{x}} - \boldsymbol{\mu})^\mathsf{T}\big)\right) \qquad (5.2.2)\\
&= \operatorname{Tr}\left(\Sigma^{-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}\right) + n(\bar{\mathbf{x}} - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\bar{\mathbf{x}} - \boldsymbol{\mu}). \qquad (5.2.3)
\end{aligned}$$

Since the first term in (5.2.3) does not depend on $\boldsymbol{\mu}$, we just need to minimize $(\bar{\mathbf{x}} - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\bar{\mathbf{x}} - \boldsymbol{\mu})$. This expression is greater than or equal tho zero given that $\Sigma^{-1}$ is positive definite, and is exactly equal to zero at $\boldsymbol{\mu} = \bar{\mathbf{x}}$. Therefore the minimum is achieved at $\boldsymbol{\mu} = \bar{\mathbf{x}}$.

To find the MLE for $\Sigma$, we plug in $\boldsymbol{\mu} = \bar{\mathbf{x}}$ as this maximizes the likelihood for any $\Sigma$. Taking (5.2.1) and (5.2.2) the goal is to maximize
$$\frac{1}{\det(\Sigma)^{n/2}}e^{-\frac{1}{2}\operatorname{Tr}\left(\Sigma^{-1}B\right)} = g(\Sigma),$$

where $g(\cdot)$ is defined in Lemma 5.2.1, with $b = n/2$ and $B = \sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}$. Lemma 5.2.1 gives that the maximum occurs at

$$\frac{1}{2^{\frac{n}{2}}}B = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}.$$

□

Similarly to the univariate Normal case, the MLE for $\Sigma$ is biased and typically one uses the unbiased version with $n-1$ in the denominator, that is

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{X}_i - \overline{\mathbf{X}})(\mathbf{X}_i - \overline{\mathbf{X}})^\mathsf{T}$$

(recall Section 4.2.4. )

### 5.2.2 Confidence regions

The following result gives the sampling distribution for $\overline{\mathbf{X}}$ and $S = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{X}_i - \overline{\mathbf{X}})(\mathbf{X}_i - \overline{\mathbf{X}})^\mathsf{T}$ for Normally distributed data.

**Proposition 5.2.3 (Multivariate version of Fisher's Theorem).** Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be an i.i.d. sample from $N_p(\boldsymbol{\mu}, \Sigma)$.

1. $\overline{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i$ is distributed $N_p(\boldsymbol{\mu}, \frac{1}{n}\Sigma)$.
2. $(n-1)S \sim \mathcal{W}_p(\Sigma, n-1)$.
3. $\overline{\mathbf{X}}$ and $S$ are independent.

*Proof.* Here is a sketch, the details of which you should complete on your own.. Since the $\mathbf{X}_i$s are i.i.d. MVN, they are jointly MVN, and therefore the sample mean (being a linear transformation of a multivariate normal) is MVN. One directly computes the expectation of $\overline{\mathbf{X}}$. For its covariance, we use the formula $\mathrm{Cov}(\mathbf{Y}) = \mathbb{E}\left[\mathbf{Y}(\mathbf{Y} - \mathbb{E}\,\mathbf{Y})^\mathsf{T}\right]$, proved in the assignment, and use the linearity of the expectation.

2. See video.
3. See video

□

**Example 5.2.4.** Let $\mathbf{X} = (X_1, X_2, X_3)^\mathsf{T} \sim N_p(\boldsymbol{\mu}, \Sigma)$ with $\boldsymbol{\mu} = (10, 5, 0)^\mathsf{T}$ and

$$\Sigma = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

Define $Y_1 = X_1 - X_2$ and $Y_2 = X_3$ and suppose we get a sample of size $n = 20$ from $\mathbf{X}$. What is the sampling distribution of $\bar{\mathbf{Y}}$?

We first note that $\mathbf{Y} = (Y_1, Y_2)^\mathsf{T} = C\mathbf{X}$, with

$$C = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore the distribution of $\mathbf{Y}$ is bivariate Normal with mean $\boldsymbol{\mu}_y = C\boldsymbol{\mu} = (5, 0)^\mathsf{T}$ and covariance $\Sigma_y = C\Sigma C^\mathsf{T} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$. Hence, the distribution of the sample mean based on $n = 20$ observations is

$$\bar{\mathbf{Y}} \sim N_2(\boldsymbol{\mu}_y, \Sigma_y/20).$$

---

The previous result tells us the sampling distribution of the MLE $(\overline{\mathbf{X}}, \frac{n-1}{n}S)$ for Normally distributed data.

In Proposition 5.1.21 we saw that, for $\overline{\mathbf{X}} \sim N_p(\boldsymbol{\mu}, \frac{1}{n}\Sigma)$, the statistic

$$n(\overline{\mathbf{X}} - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\overline{\mathbf{X}} - \boldsymbol{\mu})$$

follows a $\chi^2_p$ distribution. Hence, if $\Sigma$ were known we could use this result to obtain a $1 - \alpha$ confidence region for $\boldsymbol{\mu}$ as follows:

$$\left\{\boldsymbol{\mu} \in \mathbb{R}^p : n(\overline{\mathbf{X}} - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\overline{\mathbf{X}} - \boldsymbol{\mu}) \leq \chi^2_p(1 - \alpha)\right\}$$

When $\Sigma$ is unknown, we can replace it with an estimate and use the following result.

**Proposition 5.2.5 (Confidence regions for $\boldsymbol{\mu}$).**
Let $\mathbf{X}_1, \ldots, \mathbf{X}_n \overset{\text{iid}}{\sim} N_p(\boldsymbol{\mu}, \Sigma)$, and let $\overline{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^n \mathbf{X}_i$ and

$$S = \frac{1}{n-1}\sum_{i=1}^n (\mathbf{X}_i - \overline{\mathbf{X}})(\mathbf{X}_i - \overline{\mathbf{X}})^\mathsf{T}.$$

If $\Sigma$ is invertible, and $n \geq p + 1$, then

$$n(\overline{\mathbf{X}} - \boldsymbol{\mu})^\mathsf{T}S^{-1}(\overline{\mathbf{X}} - \boldsymbol{\mu}) \sim \frac{(n-1)p}{n-p}F_{p,n-p}$$

In particular, $1 - \alpha$ confidence region for $\boldsymbol{\mu}$ is given by the ellipsoid

$$\left\{\mathbf{y} \in \mathbb{R}^p : n(\overline{\mathbf{X}} - \mathbf{y})^\mathsf{T}S^{-1}(\overline{\mathbf{X}} - \mathbf{y}) \leq \frac{(n-1)p}{n-p}F_{p,n-p}(1 - \alpha)\right\},$$

where $F_{p,n-p}(1 - \alpha)$ is the $1 - \alpha$ quantile of an $F_{p,n-p}$ distribution.

*Proof.* Define $T^2 = \sqrt{n}(\overline{\mathbf{X}} - \boldsymbol{\mu})^\mathsf{T}S^{-1}\sqrt{n}(\overline{\mathbf{X}} - \boldsymbol{\mu})$, and note that this is the product of a $N_p(\mathbf{0}, \Sigma)$ times an independent $(\frac{1}{n-1}\mathcal{W}_p(\Sigma, n-1))^{-1}$ times the same $N_p(\mathbf{0}, \Sigma)$. By Definition 5.1.25, $T^2$ follows Hotelling's distribution with degrees of freedom $(p, n-1)$, which gives the result after applying Proposition 5.1.27. □

A basic result in probability theory is that when $n \to \infty$, $\frac{(n-1)p}{n-p} F_{p,n-p}$ converges to a $\chi^2_p$. Intuitively this makes sense, as we know by the law of large numbers and Slutsky's Lemma that $S \xrightarrow{\text{p}} \Sigma$ as $n \to \infty$, and thus applying the continuous mapping theorem we get $T^2 \xrightarrow{d} \chi^2_p$.

**Example 5.2.6.** We continue Example 5.1.22, but just use 20 draws from a bivariate normal. We compute the sample mean $\overline{\mathbf{X}} = (0.3, 0.06)$ and sample covariance

$$S = \begin{pmatrix} 1.51 & 0.97 \\ 0.97 & 1 \end{pmatrix}.$$

We compute 95% confidence regions for $\boldsymbol{\mu}$, both using the true $\Sigma$ and its estimate $S$. When $\Sigma$ is known, the confidence region is given by the ellipse

$$\{\mathbf{y} \in \mathbb{R}^2 : (\overline{\mathbf{X}} - \mathbf{y})^\intercal \Sigma^{-1} (\overline{\mathbf{X}} - \mathbf{y}) \leq \chi^2_2(0.95)/20\}.$$

When $\Sigma$ is unknown, we are looking for the ellipse

$$\{\mathbf{y} \in \mathbb{R}^p : (\overline{\mathbf{X}} - \mathbf{y})^\intercal S^{-1} (\overline{\mathbf{X}} - \mathbf{y}) \leq \frac{F_{p,18}(0.95)}{20} \times \frac{19 \times 2}{18}\}$$

These ellipses can be easily plotted in R using the `ellipse` function and indicating the cutoff point with the argument `t`. The R code is below and Figure 40 shows the results. On the upper panel we see the original draws and that both confidence regions are quite similar. On the lower panel we provide a zoom-in where we can appreciate that the 95% confidence region using $S$ is slighly larger than that for $\Sigma$ (as it's usually, though not always, the case). Both regions are centered at $\overline{\mathbf{X}}$ and contain the true value $\boldsymbol{\mu} = (0, 0)^\intercal$.

The R code below shows that the critical values for the known and unknown cases are not quite similar (0.3 vs. 0.375), as expected since $n$ is not large. In particular, this means that the confidence region without knowing the true covariance is larger to the confidence region when the true covariance is known. This difference becomes smaller if $n$ is larger.

```
op <- par(mfrow=c(1,1), mai=rep(.4,4))
plot(x,xlab='X1',ylab='X2', xlim=c(-4,4), ylim=c(-2,2))
lines(ellipse(Sest, centre=muest, t=sqrt(tf)), lwd=2)
lines(ellipse(S, centre=muest, t=sqrt(tchi)),col=2,lty=2, lwd=2)
points(muest[1],muest[2], pch=15, cex=1.5)
points(mu[1],mu[2], pch=17, cex=1.5, col=4)
#text(muest[1],muest[2],'Sample mean',pos=1,cex=1)
#text(mu[1],mu[2],'True mean',pos=1,cex=1,col='blue')
legend('topright', legend=c('True mean', 'Sample mean'),
       pch=c(17, 15), col=c(4,1))
par(op)
```

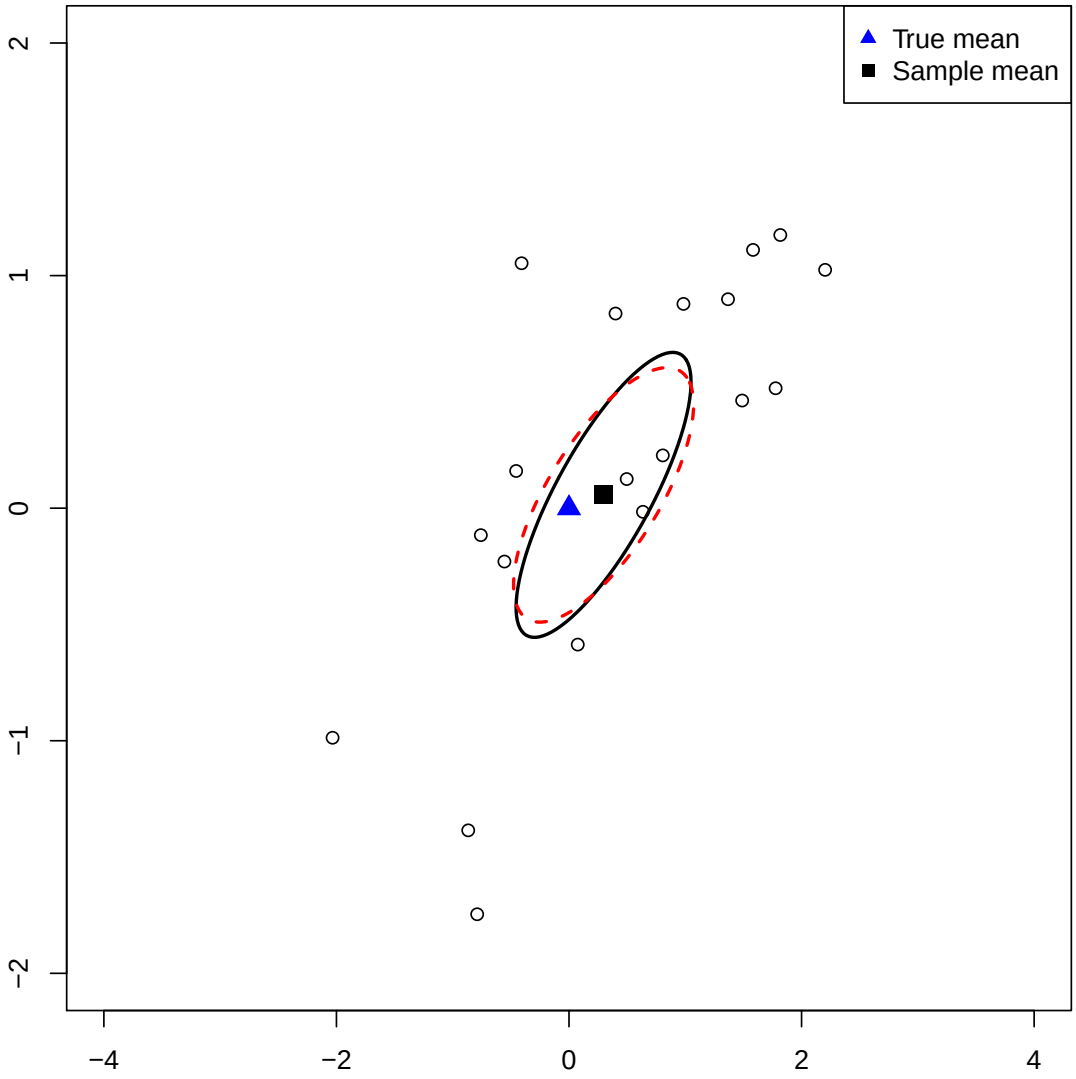



Figure 40: Bivariate Normal data and 95% confidence region using true (dashed red) and estimated covariance (solid black).

Proposition 5.2.5 tells us how to obtain confidence regions for the mean of a single population. Often we will have data arising from several populations, in which case we may be interested in obtaining confidence regions for each group. In general, we can simply apply Proposition 5.2.5 to each group separately. However, when the covariances in each group are equal (or in practice, similar enough) and the sample size per group is limited, it may be preferrable to obtain an overall estimate for $\Sigma$ across all groups. In that case the following result applies.

**Proposition 5.2.7 (Confidence regions for $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$).**
Let $\overline{\mathbf{X}}_1, \ldots, \overline{\mathbf{X}}_K$ be the sample means from $K$ independent multivariate Normal populations with means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, and common covariance $\Sigma$. Let

$$S_p = \frac{1}{N-K} \sum_{i=1}^{K} (n_i - 1) S_i,$$

where $n_1, \ldots, n_K$ and $S_1, \ldots, S_K$ are the sample sizes and sample covariance matrices in each group (respectively) and $N = \sum_{i=1}^{K} n_i$. Then

$$n_i (\overline{\mathbf{X}}_i - \boldsymbol{\mu}_i)^\mathsf{T} S_p^{-1} (\overline{\mathbf{X}}_i - \boldsymbol{\mu}_i) \sim \frac{(N-K)p}{N-K-p+1} F_{p,N-K-p+1}$$

In particular, A $1-\alpha$ confidence region for $\boldsymbol{\mu}_i$ is given by the ellipse

$$\left\{ \mathbf{y}_i \in \mathbb{R}^p : n_i (\overline{\mathbf{X}}_i - \mathbf{y}_i)^\mathsf{T} S_p^{-1} (\overline{\mathbf{X}}_i - \mathbf{y}_i) \leq \frac{(N-K)p}{N-K-p+1} F_{p,N-K-p+1}(1-\alpha) \right\}$$

*Proof.* Because of the additive property of independent Wishart distributions, the pooled sample covariance $S_p$ follows a Wishart with $N-K$ degrees of freedom. Hence the result follows immediately from the proof of Proposition 5.2.5, by simply noting that we have the product of a $N_p(\mathbf{0}, \Sigma)$ times the inverse of an independent $\frac{1}{N-K}\mathcal{W}_p(\Sigma, N-K)$ times the same $N_p(\mathbf{0}, \Sigma)$. □

### 5.2.3 Asymptotics of the Sample Mean

We have see that $\overline{\mathbf{X}}$ is the MLE of the mean of a MVN random vector. However (provided $\Sigma$ is finite), $\overline{\mathbf{X}}$ may be a sensible estimator for non-normally distributed data as well, as it estimates the population mean consistently as $n \to \infty$. Also, when $n \to \infty$ then $\overline{\mathbf{X}}$ is approximately Normal even when the individual $\mathbf{X}_i$'s are not Normally distributed.

**Proposition 5.2.8 (Law of Large Numbers and Central Limit Theorem).**
Let $\mathbf{X}_1, \ldots, \mathbf{X}_n \overset{\text{iid}}{\sim} \mathbf{X}$ be an i.i.d. sample from a multivariate distribution.

1. If $\mathbb{E}|\mathbf{X}| < \infty$, then

$$\overline{\mathbf{X}} \xrightarrow{\text{a.s.}} \boldsymbol{\mu}, \quad \text{as } n \to \infty,$$

where $\boldsymbol{\mu} = \mathbb{E}\,\mathbf{X}$.

2. If $\mathbb{E}|\mathbf{X}|^2 < \infty$, then

$$\sqrt{n}(\overline{\mathbf{X}} - \boldsymbol{\mu}) \xrightarrow{d} N_p(0, \Sigma), \quad \text{as } n \to \infty,$$

where $\Sigma = \mathrm{Cov}(\mathbf{X})$.

*Proof.* Part (1) follows from applying the univariate Strong Law of Large Numbers for each element in $\overline{\mathbf{X}}$.

We do not prove Part (2) here, but the result follows from seeing that the characteristic function of $\overline{\mathbf{X}}$ converges to that of a multivariate Normal (analogously to the proof for the univariate Central Limit Theorem). □

## 5.3 Hypothesis testing

We now focus on testing hypotheses, that is on determining whether a given parameter value (or function of parameter values) is likely to have generated the observed data. These multivariate tests generalize the usual one-sample t-test, two-sample t-test and F-test (ANOVA) to the case where there are several response variables.

The main difference between univariate and multivariate hypothesis tests is that the former focus on a single variable, while the latter perform an overall comparison that takes all variables into account. As an example suppose we have two variables $X$ and $Y$, and that we want to compare their expected values between two groups. One option is to use univariate tests (*e.g.* t-tests) to compare means for each variable separately. Some limitations and advantages of this univariate approach are:

1. Unless we somehow adjust for multiple comparisons, the family-wise type-I error rate for the test will be greater than the desired $\alpha$.

2. Exactly which multiple comparisons adjustment is more adequate depends on the dependence between $X$ and $Y$.

3. When there truly are differences between groups both for $X$ and $Y$, a multivariate analysis will in general have greater power to detect them. For instance, the multivariate test can achieve statistical significance even when both univariate tests do not.

4. Conversely, when only some variables exhibit differences between groups a univariate approach may have higher power than a multivariate one.

5. The univariate approach may be easier to explain to non-experts.

Recall that for univariate tests there is a direct relationship between confidence intervals and P-values. As we shall see below, the same kind of relationship holds for multivariate tests, where P-values are directly related to confidence regions.

The methods we see below assume the data have an MVN distribution, thanks to the Central Limit Theorem they can also be used whenever $n$ is moderately large.

**Remark 5.3.1 (Rejecting and Accepting Hypotheses).**
It is bad practice to say "we reject the null" without saying the level at which you reject the null. Not rejecting the null at a specific level does not imply that "we accept the null hypothesis". Read the paper called *Retire Statistical Significance* (available on Moodle) for advice on best practice.

### 5.3.1 Test 1 multivariate Normal mean

Let us start by considering the 1-sample problem. Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be independent draws from a $N_p(\boldsymbol{\mu}, \Sigma)$ distribution, and suppose we are interested in testing

$$H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0 \text{ vs. } H_1 : \boldsymbol{\mu} \neq \boldsymbol{\mu}_0,$$

where $\boldsymbol{\mu}_0$ is a known constant.

Given what we know from Proposition 5.2.5, a natural test statistic is

$$T^2 = n(\overline{\mathbf{X}} - \boldsymbol{\mu}_0)^\mathsf{T} S^{-1} (\overline{\mathbf{X}} - \boldsymbol{\mu}_0),$$

which is called Hotelling's $T^2$ test. It's named after Harold Hotelling, who proposed it as early as 1931 as a generalization of the squared Student's t-statistic for univariate tests

$$t^2 = n \frac{(\bar{X} - \mu_0)^2}{\hat{\sigma}^2},$$

where $\hat{\sigma}^2$ is the sample variance.

**Proposition 5.3.2 (T-squared Test).** Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be independent draws from a $N_p(\boldsymbol{\mu}, \Sigma)$. Assume that $\Sigma$ is invertible and that $n \geq p + 1$. Under $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0$,

$$T^2 = n(\overline{\mathbf{X}} - \boldsymbol{\mu}_0)^\mathsf{T} S^{-1} (\overline{\mathbf{X}} - \boldsymbol{\mu}_0) \sim \frac{(n-1)p}{n-p} F_{p,n-p}.$$

The result a direct consequence of Proposition 5.2.5 and the definition of Hotelling's $T^2$ distribution. Notice that any $T^2 > \frac{(n-1)p}{n-p} F_{p,n-p}(1-\alpha)$ (for which $H_0$ is rejected at level $\alpha$) falls outside the confidence region given by the ellipse

$$\{\mathbf{z} : n(\mathbf{z} - \boldsymbol{\mu}_0)^\mathsf{T} S^{-1} (\mathbf{z} - \boldsymbol{\mu}_0) = \frac{(n-1)p}{n-p} F_{p,n-p}(1-\alpha)\}.$$

That is, testing $H_0$ at level $\alpha$ using $T^2$ is equivalent to checking whether $\overline{\mathbf{X}}$ falls inside the $1-\alpha$ confidence region.

**Example 5.3.3.** $n = 15$ university students were asked to evaluate the importance (in a scale from 0 to 100) of 4 items for a module to be successful: teacher explaining well ($X_1$), teacher being enthusiastic ($X_2$), material being useful ($X_3$) and material being interesting ($X_4$).

| $Y_1$ | $Y_2$ |
|---|---|
| 0 | 5 |
| 10 | 30 |
| 10 | 0 |
| 0 | 0 |
| 40 | 0 |
| 20 | 15 |
| −30 | 0 |
| −2 | −3 |
| −10 | −30 |
| 30 | −35 |
| 0 | −30 |
| 5 | 48 |
| 10 | −10 |
| 10 | −15 |
| 0 | −20 |

Table 1: Importance of teacher and material for a module to be successful, evaluated by 20 students in a scale 0–100. $Y_1$: difference between importance of teacher explaining well and his/her being enthusiastic. $Y_2$: difference between importance of material being useful and its being interesting.

A university official suspects that, on the average, students give the same rating to the two teacher-related questions, and that the same is true for the course material-related questions. With this in mind, he computes the differences $Y_1 = X_1 - X_2$ and $Y_2 = X_3 - X_4$, which are displayed in Table 1, and wishes to test the null hypothesis

$$H_0 : \boldsymbol{\mu} = \begin{pmatrix} \mathbb{E}\, Y_1 \\ \mathbb{E}\, Y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Let us perform this test, assuming $(Y_1, Y_2)$ follow a bivariate Normal. The sample mean and covariance are

$$\overline{\mathbf{Y}} = \begin{pmatrix} 6.2 \\ -3.0 \end{pmatrix}; S = \begin{pmatrix} 268.03 & 8.93 \\ 8.93 & 505.57 \end{pmatrix}$$

and inverse covariance

$$S^{-1} = \begin{pmatrix} 3.7{\times}10^{-3} & -6.59{\times}10^{-5} \\ -6.59{\times}10^{-5} & 1.98{\times}10^{-3} \end{pmatrix}.$$

We see that there is little correlation between $Y_1$ and $Y_2$. The $T^2$ statistic is equal to $n\overline{\mathbf{Y}}^{\mathsf{T}} S^{-1}\overline{\mathbf{Y}} = 2.456$, hence the P-value is given by the tail area of $T^2 \frac{n-p}{(n-1)p} = 1.140$ under a $F_{2,13}$ distribution, and is equal to 0.350.

We compare these results with two separate t-tests for $Y_1$ and $Y_2$. The p-values are 0.164 and 0.613 (respectively).

Figure 41 shows the 95% confidence region for $\boldsymbol{\mu}$, which is an ellipse containing $\boldsymbol{\mu}_0$. As we saw before, the 95% confidence ellipse will contain $\boldsymbol{\mu}_0$ whenever the $T^2$

P-value is above 0.05. The figure also indicates the univariate confidence intervals (blue dashed lines) for each mean, obtained through the usual (univariate) formula

$$\bar{Y}_i \pm t_{n-1}(1-\frac{\alpha}{2})\sqrt{\frac{s_{ii}}{n}}$$

for $i = 1, 2$, where $t_{n-1}(1-\frac{\alpha}{2})$ is the $1-\frac{\alpha}{2}$ quantile of Students t distribution with $n-1$ degrees of freedom. We appreciate that the confidence intervals define a square in space which does not achieve 95% joint coverage. In order for the square to achieve 95% coverage, the univariate intervals should be widened to take into account that we're considering two variables simultaneously. We will not discuss how this widening could be done, but the intuitive idea should be clear from the picture. If you are interested in this, search for "multiple testing".

```
x <- read.csv('~/st323/data/course_happy.csv',header=TRUE)
x <- x[c(1,2,3,8)]
x <- x[,c('Teacher_explaining','Material_useful'),] -
    x[,c('Teacher_enthusiastic','Material_stimulating')]
colnames(x) <- c('ExplainVsEnthu','UsefulVsStim')
x <- head(x,n=15)

m <- matrix(colMeans(x),ncol=1)
S <- cov(x)
Sinv <- solve(S)
n <- nrow(x); p <- ncol(x)

T2 <- n * t(m) %*% Sinv %*% m
tf <- qf(.95,df1=p,df2=n-p) * (n-1)*p/(n-p)
pval <- 1 - pf(T2 * (n-p)/((n-1)*p), df1=p,df2=n-p)
pval
```

```
##           [,1]
## [1,] 0.3496542
```

```
ttest1 <- t.test(x[,1]); ttest2 <- t.test(x[,2])
ci1 <- ttest1$conf.int; ci2 <- ttest2$conf.int

ttest1$p.value; ttest2$p.value
```

```
## [1] 0.1645539
## [1] 0.6134007
```

```
ci1; ci2
```

```
## [1] -2.866278 15.266278
## attr(,"conf.level")
## [1] 0.95
## [1] -15.451731   9.451731
## attr(,"conf.level")
## [1] 0.95
```

```r
ci1.bonferroni <- t.test(x[,1], conf.level=0.975)$conf.int
ci2.bonferroni <- t.test(x[,2], conf.level=0.975)$conf.int

library(ellipse)
plot(ellipse(S/n, centre=m, t=sqrt(tf)), type='l',xlab='Y1',ylab='Y2')
abline(v=ci1,col='blue',lty=2)
abline(v=ci1.bonferroni,col='red',lty=4)
abline(h=ci2,col='blue',lty=2)
abline(h=ci2.bonferroni,col='red',lty=4)
points(m[1],m[2],pch=16,cex=2)
text(m[1],m[2],pos=1,'Sample mean',cex=1.3)
points(0,0,pch=17,cex=2,col='darkgray')
text(0,0,'mu0',cex=1.3,col='darkgray',pos=3)


S0 <- t(as.matrix(x)) %*% as.matrix(x) / (n-1)

l <- (det(S) / det(S0))^(n/2)
l^(2/n)

## [1] 0.8507277

(1+T2/(n-1))^(-1)

##           [,1]
## [1,] 0.8507277
```

---

**Example 5.3.4.** Related to the previous example, $n = 63$ students were asked to evaluate the importance of clarity and enthusiasm when a teacher presents material. Suppose we wish to test whether students at the university would rate these two issues with an average score of 75. That is, given the sample of $n = 63$ students, we want to test the null hypothesis $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0 = (75, 75)^\mathsf{T}$.

To do the calculations, we are told that $\bar{\mathbf{Y}} = (88.3, 77.0)^\mathsf{T}$ and

$$S = \begin{pmatrix} 132.8 & 64.9 \\ 64.9 & 224.8 \end{pmatrix}.$$

We compute $T^2 = n(\bar{\mathbf{Y}} - \boldsymbol{\mu}_0)^\mathsf{T} S^{-1} (\bar{\mathbf{Y}} - \boldsymbol{\mu}_0) = 90.855$, hence the P-value is given by the tail area of $T^2 \frac{n-p}{(n-1)p} = 44.695$ under a $F_{2,61}$ distribution, which is equal to $1.115 \times 10^{-12}$.

We compare these results with separate univariate t-tests. The t-test for $Y_1$ returns a P-value of $3.61 \times 10^{-13}$, and that for $Y_2$ is 0.2825. That is, there is strong evidence against $\mu_1 = 75$ but no so much against $\mu_2 = 75$.

Figure 42 shows the 95% confidence ellipse for $\boldsymbol{\mu}$ (solid black) and the univariate 95% confidence intervals. The hypothesized $\boldsymbol{\mu}_0 = (75, 75)^\mathsf{T}$ lies far away from the

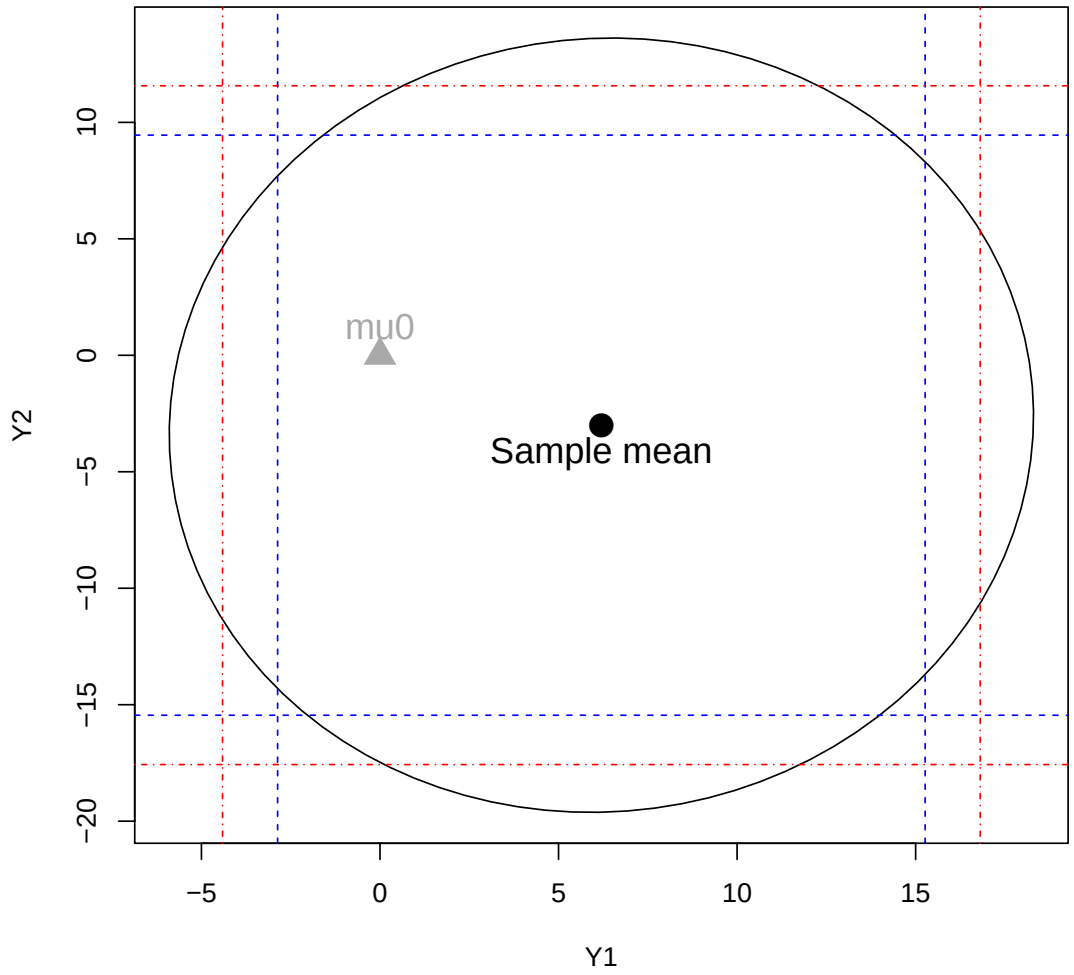



Figure 41: 95% confidence region for module evaluation data (solid black ellipse), and univariate 95% confidence intervals (blue dashed lines). The dash-dotted red lines indicated univariate 95% confidence intervals that have been adjusted for multiplicity (using Bonferroni's correction). The sample mean and hypothesized $\boldsymbol{\mu}_0 = (0, 0)^{\mathsf{T}}$ are also indicated.

ellipse, in agreement with the small P-value in the bivariate test. We note that the shape of the ellipse reflects the positive correlation between $Y_1$ and $Y_2$, namely $64.9/\sqrt{132.8 \times 224.8} = 0.376$. With higher degrees of correlation, the shape of the ellipse will differ even more from the rectangle given by the univariate confidence intervals.

```
x <- read.csv('~/st323/data/course_happy.csv',header=TRUE)
x <- x[,1:2]
#x <- x[c(1,2,3,8)]
#x <- x[,c('Teacher_explaining','Material_useful'),] - x[,c('Teacher_enthusiastic
#colnames(x) <- c('ExplainVsEnthu','UsefulVsStim')
#x <- head(x,n=15)

m <- matrix(colMeans(x),ncol=1)
mu0 <- c(75,75)
S <- cov(x)
Sinv <- solve(S)
n <- nrow(x); p <- ncol(x)

T2 <- n * matrix(m-mu0,nrow=1) %*% Sinv %*% matrix(m-mu0,ncol=1)
tf <- qf(.95,df1=p,df2=n-p) * (n-1)*p/(n-p)
pval <- 1 - pf(T2 * (n-p)/((n-1)*p), df1=p,df2=n-p)
pval

##              [,1]
## [1,] 1.11533e-12

ttest1 <- t.test(x[,1],mu=mu0[1]); ttest2 <- t.test(x[,2],mu=mu0[2])
ci1 <- ttest1$conf.int; ci2 <- ttest2$conf.int

ttest1$p.value; ttest2$p.value

## [1] 3.608413e-13
## [1] 0.2825241

as.numeric(ci1); as.numeric(ci2)

## [1] 85.43101 91.23566
## [1] 73.27197 80.82327

plot(ellipse(S/n, centre=m, t=sqrt(tf)), type='l',xlab='Y1',
     ylab='Y2', xlim=c(72,92),ylim=c(72,92))
abline(v=ci1,col='blue',lty=2)
abline(h=ci2,col='blue',lty=2)
points(m[1],m[2],pch=16,cex=2)
text(m[1],m[2],pos=1,'Sample mean',cex=1.3)
points(mu0[1],mu0[2],pch=17,cex=2,col='darkgray')
```

```r
text(mu0[1],mu0[2],'mu0',cex=1.3,col='darkgray',pos=3)

x2 <- t(t(x)-mu0)
S0 <- t(as.matrix(x2)) %*% as.matrix(x2) / (n-1)

l <- (det(S) / det(S0))^(n/2)
l^(2/n)

## [1] 0.4056132

(1+T2/(n-1))^(-1)

##           [,1]
## [1,] 0.4056132

-2*log(l)

## [1] 56.84839

1-pchisq(-2*log(l),df=2)

## [1] 4.524159e-13
```

---

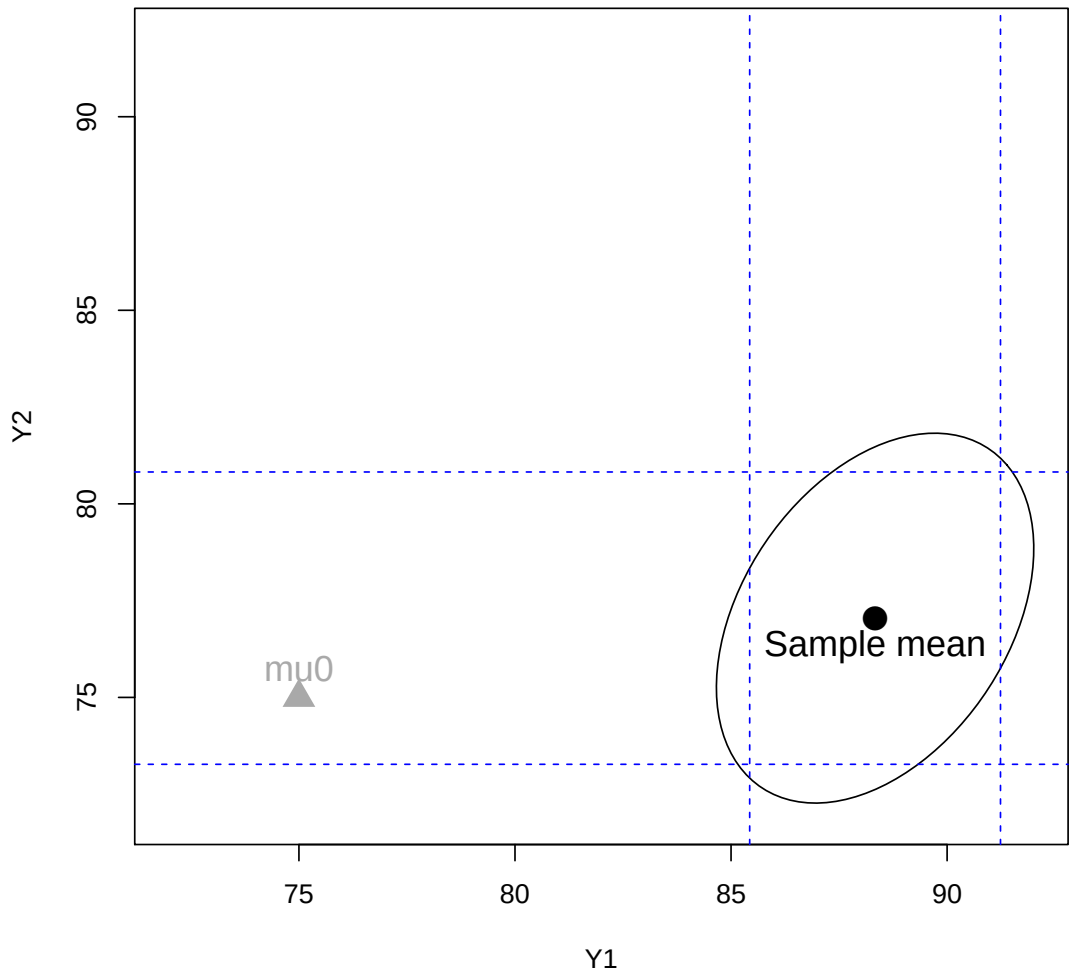



Figure 42: 95% confidence ellipse for $(\mu_1, \mu_2)$: mean scores for importance of teacher explaining clarity $(Y_1)$ and enthusiasm $(Y_2)$. Blue dashed lines: univariate 95% confidence intervals. The sample mean and hypothesized $\boldsymbol{\mu}_0 = (75, 75)^\mathsf{T}$ are also indicated.

### 5.3.2 Likelihood Ratio Tests

So far we have based all our hypothesis tests on the test statistic $T^2$, but naturally there are other possible choices. As an alternative, one may consider a Likelihood Ratio Test, which provides a general recipe for testing hypotheses.

**Definition 5.3.5.** Let $\mathbf{X}_1, \ldots, \mathbf{X}_n \overset{\text{iid}}{\sim} N_p(\boldsymbol{\mu}, \Sigma)$, with $n > p$ and $\Sigma$ invertible. Let $(\boldsymbol{\mu}_0, \hat{\Sigma}_0)$ be the MLE for $(\boldsymbol{\mu}, \Sigma)$ under $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0$ and $(\overline{\mathbf{X}}, \hat{\Sigma})$ be that under $H_1 : \boldsymbol{\mu} \neq \boldsymbol{\mu}_0$. The likelihood ratio test is based on setting a critical value for the statistic

$$\Lambda = \frac{f(\mathbf{X}_1, \ldots, \mathbf{X}_n \mid \boldsymbol{\mu}_0, \hat{\Sigma}_0)}{f(\mathbf{X}_1, \ldots, \mathbf{X}_n \mid \overline{\mathbf{X}}, \hat{\Sigma})}.$$

The critical value for this test is found by analysing the asymptotic behaviour of $\Lambda$ under the null hypothesis.

**Proposition 5.3.6.** In the setting of Definition 5.3.5, we have $\hat{\Sigma}_0 = \frac{1}{n}\sum_{i=1}^n (\mathbf{X}_i - \boldsymbol{\mu}_0)(\mathbf{X}_i - \boldsymbol{\mu}_0)^\mathsf{T}$ and

$$\Lambda = \left( \frac{\det(\hat{\Sigma})}{\det(\hat{\Sigma}_0)} \right)^{n/2}$$

Moreover, under $H_0$,

$$-2\log(\Lambda) \xrightarrow{d} \chi^2_p.$$

*Proof.* Plugging $\overline{\mathbf{X}}$ and $\hat{\Sigma}$ into the multivariate Normal likelihood, the denominator in $\Lambda$ simplifies to

$$\frac{1}{(2\pi)^{\frac{np}{2}} \det(\hat{\Sigma})^{\frac{n}{2}}} e^{-\frac{1}{2}\operatorname{Tr}\left(\hat{\Sigma}^{-1}\sum_{i=1}^n (\mathbf{X}_i - \overline{\mathbf{X}})^\mathsf{T}(\mathbf{X}_i - \overline{\mathbf{X}})\right)} = \frac{1}{(2\pi)^{\frac{np}{2}} \det(\hat{\Sigma})^{\frac{n}{2}}} e^{-\frac{1}{2}np}.$$

To evaluate the numerator in $\Lambda$ we need to find $\hat{\Sigma}_0$, the MLE for $\Sigma$ when $\boldsymbol{\mu} = \boldsymbol{\mu}_0$. The likelihood function with $\boldsymbol{\mu} = \boldsymbol{\mu}_0$ is

$$\frac{1}{(2\pi)^{\frac{np}{2}} \det(\Sigma)^{\frac{n}{2}}} e^{-\frac{1}{2}\operatorname{Tr}\left(\Sigma^{-1}\sum_{i=1}^n (\mathbf{X}_i - \boldsymbol{\mu}_0)^\mathsf{T}(\mathbf{X}_i - \boldsymbol{\mu}_0)\right)}, \tag{5.3.1}$$

which using Lemma 5.2.1 is maximized for $\hat{\Sigma}_0 = \frac{1}{n}\sum_{i=1}^n (\mathbf{X}_i - \boldsymbol{\mu}_0)^\mathsf{T}(\mathbf{X}_i - \boldsymbol{\mu}_0)$. Plugging $\hat{\Sigma}_0$ into (5.3.1) we obtain

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_n \mid \boldsymbol{\mu}_0, \hat{\Sigma}_0) = \frac{1}{(2\pi)^{\frac{np}{2}} \det(\hat{\Sigma}_0)^{\frac{n}{2}}} e^{-\frac{1}{2}np}.$$

Therefore $\Lambda = (\det(\hat{\Sigma})/\det(\hat{\Sigma}_0))^{n/2}$, which proves the first part of the result.

The proof of the asymptotic $\chi^2_p$ distribution is non-examinable, but note that the proof is based on the Normal distribution of $\overline{\mathbf{X}}$. Hence, by virtue of the Central Limit Theorem, in many instances the result can be extended for non-Normally distributed data. □

Note that $\Lambda$ is not affected by using $n-1$ or $n$ in the denominator of $\hat{\Sigma}$ and $\hat{\Sigma}_0$, as this constant cancels in the ratio

$$\frac{\det(S)}{\det(\hat{\Sigma}_0)} = \frac{\frac{1}{n^p}\det\left(\sum_{i=1}^n (\mathbf{X}_i - \overline{\mathbf{X}})(\mathbf{X}_i - \overline{\mathbf{X}})^\mathsf{T}\right)}{\frac{1}{n^p}\det\left(\sum_{i=1}^n (\mathbf{X}_i - \boldsymbol{\mu}_0)(\mathbf{X}_i - \boldsymbol{\mu}_0)^\mathsf{T}\right)}.$$

The reason why $\Lambda$ is an appealing test statistic is that when testing two point hypotheses $H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0$ versus $H_1 : \boldsymbol{\mu} = \boldsymbol{\mu}_1$ it achieves the highest power amongst all level $\alpha$ tests. Of course, in our context the alternative is not a point hypothesis so the optimality property does not hold anymore, but it seems reasonable to expect that $\Lambda$ will still achieve good power. For Normally distributed data, it turns out that $\Lambda$ is equivalent to $T^2$. The practical consequence is that we do not need to use the asymptotic $\chi^2_p$ distribution for $\Lambda$, but simply use the exact distribution for $T^2$.

**Proposition 5.3.7 (LRT and Hotelling's test).** Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be independent draws from a $N_p(\boldsymbol{\mu}, \Sigma)$, $T^2$ Hotelling's statistic (Proposition 5.3.2) and $\Lambda$ the likelihood ratio (Proposition 5.3.6). Then

$$\Lambda^{2/n} = \left(1 + \frac{T^2}{(n-1)}\right)^{-1}$$

*Proof.* Left as an exercise. □

**Example 5.3.8.** We continue with the module evaluation data in Example 5.3.3. As discussed, for that data we observe $T^2 = 2.456$ with corresponding exact P-value=0.349.

The MLE for the covariance under the null is

$$\hat{\Sigma}_0 = \begin{pmatrix} 309.21 & -11.0 \\ -11.0 & 515.21 \end{pmatrix}.$$

Therefore, $\det(S) = 135427.9$, $\det(\hat{\Sigma}_0) = 159190.6$ and $\Lambda = 0.297$.

We can easily check that $\Lambda^{2/n} = 0.8507 = \left(1 + \frac{T^2}{n-1}\right)^{-1}$, as predicted by the theory.

Alternatively, one could compute an approximate P-value from the asymptotic $\chi^2_2$ distribution of $-2\log(\Lambda)$. In our dataset $-2\log(\Lambda) = 2.425$, with a corresponding P-value=0.297. Both P-values are similar, and provide essentially the same conclusions.

---

**Example 5.3.9.** We continue Example 5.3.4 with $n = 63$ students, now using a likelihood ratio test for $H_0 : \boldsymbol{\mu} = (75, 75)'$. In order to carry out the calculations, we're told that the MLE for $\Sigma$ under $H_0$ is

$$\hat{\Sigma}_0 = \begin{pmatrix} 313.5 & 92.7 \\ 92.7 & 229.0 \end{pmatrix}.$$

Therefore, $\det(\hat{\Sigma}_0) = 63199.36$ and $\det(S) = 25634.49$, so that

$$\Lambda = \left( \frac{\det(S)}{\det(\hat{\Sigma}_0)} \right)^{n/2} = 4.52 \times 10^{-13}.$$

To determine the (asymptotic) P-value we compute $-2\log(\Lambda) = 56.848$ and obtain $P(\chi^2_2 \geq 56.848) = 4.524 \times 10^{-13}$. Recall that with the $T^2$ statistic we found $1.115 \times 10^{-12}$, which in practice amounts to the same overwhelming evidence against $H_0$.

---

### 5.3.3 Compare 2 multivariate Normal means

The methods for testing a single mean vector extend naturally to comparing mean vectors between two populations when the covariances in the two populations are equal. Otherwise, we must rely on asymptotic distributions. The following result summarizes the two cases.

**Proposition 5.3.10 (2-sample Hotelling's test).** Let $\mathbf{X}_1, \ldots, \mathbf{X}_{n_1}$ be an i.i.d. sample from $N_p(\boldsymbol{\mu}_1, \Sigma_1)$ and $\mathbf{Y}_1, \ldots, \mathbf{Y}_{n_2}$ an i.i.d. sample from $N_p(\boldsymbol{\mu}_2, \Sigma_2)$, and assume that the $\mathbf{X}_i$s and $\mathbf{Y}_j$s are independent. Denote by $\overline{\mathbf{X}}, \overline{\mathbf{Y}}$ the sample means and $S_1, S_2$ the sample covariance matrices ($n_1 - 1$ and $n_2 - 1$ in the denominator).

1. Suppose that $\Sigma_1 = \Sigma_2$ and let $S_p = \frac{(n_1-1)S_1 + (n_2-1)S_2}{n_1+n_2-2}$. Then under $H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ the test statistic

$$T^2 = (\overline{\mathbf{X}} - \overline{\mathbf{Y}})^\mathsf{T} \left[ \left( \frac{1}{n_1} + \frac{1}{n_2} \right) S_p \right]^{-1} (\overline{\mathbf{X}} - \overline{\mathbf{Y}})$$

follows the distribution

$$\frac{(n_1 + n_2 - 2)p}{(n_1 + n_2 - p - 1)} F_{p, n_1+n_2-p-1}.$$

2. Suppose that $\Sigma_1 \neq \Sigma_2$. If $n_1, n_2 \to \infty$ such that $n_1/(n_1 + n_2) \to c \in (0, 1)$, then under $H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$, we have

$$T^2 = (\overline{\mathbf{X}} - \overline{\mathbf{Y}})^\mathsf{T} \left( \frac{S_1}{n_1} + \frac{S_2}{n_2} \right)^{-1} (\overline{\mathbf{X}} - \overline{\mathbf{Y}}) \xrightarrow{d} \chi^2_p$$

as $n_1, n_2 \to \infty$.

*Proof.* To prove the first part, we start by noting that

$$\begin{pmatrix} \overline{\mathbf{X}} \\ \overline{\mathbf{Y}} \end{pmatrix} \sim N_p \left( \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \Sigma_1/n_1 & \mathbf{0} \\ \mathbf{0} & \Sigma_2/n_2 \end{pmatrix} \right),$$

because of independence between $\mathbf{X}_1, \ldots, \mathbf{X}_{n_1}$ and $\mathbf{Y}_1, \ldots, \mathbf{Y}_{n_2}$. Now define the matrix

$$C = \begin{pmatrix} 1 & 0 & \ldots & 0 & -1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & 0 & -1 & \ldots & 0 \\ & & \ldots & & & & & \\ 0 & 0 & \ldots & 1 & 0 & 0 & \ldots & -1 \end{pmatrix} = \left(I_p | -I_p\right),$$

where $I_p$ is the $p \times p$ identity matrix. Then clearly $C(\overline{\mathbf{X}}, \overline{\mathbf{Y}})^\mathsf{T} = \overline{\mathbf{X}} - \overline{\mathbf{Y}}$ and therefore by Proposition 5.1.6 we have that $\overline{\mathbf{X}} - \overline{\mathbf{Y}}$ is multivariate Normal with mean $C(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)^\mathsf{T} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ and covariance

$$C \begin{pmatrix} \frac{1}{n_1}\Sigma_1 & \mathbf{0} \\ \mathbf{0} & \frac{1}{n_2}\Sigma_2 \end{pmatrix} C^\mathsf{T} = \left(\tfrac{1}{n_1}\Sigma_1 | \tfrac{-1}{n_2}\Sigma_2\right) C^\mathsf{T} = \frac{1}{n_1}\Sigma_1 + \frac{1}{n_2}\Sigma_2.$$

Therefore when $\Sigma_1 = \Sigma_2$ and $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ we get

$$\overline{\mathbf{X}} - \overline{\mathbf{Y}} \sim N_p\left(\mathbf{0}, \left(\frac{1}{n_1} + \frac{1}{n_2}\right)\Sigma\right).$$

We also note that $(n_1 - 1)S_1 + S_2(n_2 - 1)$ follows a $\mathcal{W}_p(\Sigma, n_1 + n_2 - 2)$ distribution, since its summands are independent and $\mathcal{W}_p(\Sigma, n_i - 1)$ distributed (Proposition 5.1.24). Therefore $S_p = ((n_1 - 1)S_1 + S_2(n_2 - 1))/(n_1 + n_2 - 2)$ is a $\mathcal{W}_p(\Sigma, n_1 + n_2 - 2)$ divided by its degrees of freedom, so that

$$\begin{aligned} T^2 &= \left(\frac{1}{n_1} + \frac{1}{n_2}\right)^{-1/2} (\overline{\mathbf{X}} - \overline{\mathbf{Y}})^\mathsf{T} (S_p)^{-1} \left(\frac{1}{n_1} + \frac{1}{n_2}\right)^{-1/2} (\overline{\mathbf{X}} - \overline{\mathbf{Y}}) \\ &= N_p(\mathbf{0}, \Sigma)^\mathsf{T} \left(\frac{\mathcal{W}_p(\Sigma, n_1 + n_2 - 2)}{n_1 + n_2 - 2}\right)^{-1} N_p(\mathbf{0}, \Sigma). \end{aligned}$$

Since $S_p$ is independent of $\overline{\mathbf{X}} - \overline{\mathbf{Y}}$, by definition of Hotelling's $T^2$ distribution,

$$T^2 \sim \frac{(n_1 + n_2 - 2)p}{n_1 + n_2 - p - 1} F_{p, n_1 + n_2 - 1 - p}.$$

We sketch the proof of the second part (non-examinable). Let $n = n_1 + n_2$. First, show that

$$\sqrt{n}(\overline{\mathbf{X}} - \mu_1) \xrightarrow{d} N_p(0, \Sigma_1/c), \qquad \sqrt{n}(\overline{\mathbf{Y}} - \mu_2) \xrightarrow{d} N_p(0, \Sigma_2/(1-c)),$$

and since the $\mathbf{X}$s and $\mathbf{Y}$s are independent, by the continuous mapping theorem we get

$$\sqrt{n}(\overline{\mathbf{X}} - \overline{\mathbf{Y}}) \xrightarrow{d} N_p(0, \Sigma_1/c + \Sigma_2/(1-c)).$$

Next, show that

$$n\left(S_1/n_1 + S_2/n_2\right) \xrightarrow{\mathrm{p}} \Sigma_1/c + \Sigma_2/(1-c).$$

This can be done using Slutsky's Lemma. The continuous mapping theorem therefore implies that

$$\begin{aligned} (\overline{\mathbf{X}} - \overline{\mathbf{Y}})^\mathsf{T} \left(S_1/n_1 + S_2/n_2\right)^{-1} (\overline{\mathbf{X}} - \overline{\mathbf{Y}}) &= \sqrt{n}(\overline{\mathbf{X}} - \overline{\mathbf{Y}})^\mathsf{T} \left(n[S_1/n_1 + S_2/n_2]\right)^{-1} \sqrt{n}(\overline{\mathbf{X}} - \overline{\mathbf{Y}}) \\ &\xrightarrow{d} \mathbf{Z}^\mathsf{T}(\Sigma_1/c + \Sigma_2/(1-c))^{-1}\mathbf{Z} \\ &\overset{d}{=} \chi^2_p, \end{aligned}$$

where $\mathbf{Z} \sim N_p(0, \Sigma_1/c + \Sigma_2/(1-c))$. □

In practice it is hard to determine whether $\Sigma_1$ and $\Sigma_2$ are similar enough to use part (1) of Proposition 5.3.10. There are available tests for the equality between covariance matrices, but these usually depend critically on the assumption of multivariate Normality. An alternative is to use permutation-based approaches, but we shall not pursue this here. For our purposes, we will assume $\Sigma_1 = \Sigma_2$ unless there is overwhelming evidence to the contrary, and the assumption usually delivers reasonable results in practice. In R the two-sample Hotelling test is available, for instance, in function `hotelling.test` of package `Hotelling`.

**Example 5.3.11.** We revisit the example with importance scores assigned by students in order to be happy with a course. Let $X_1$ be the importance assigned to the teacher explaining well, $X_2$ to the teacher being enthusiastic and $X_3$ of having good lecture notes. We wish to compare the mean scores between two groups of students ($3^{rd}$ year *versus* MSc), based on samples of $n_1 = 34$ and $n_2 = 21$ students, assuming that scores are multivariate Normal distributed.

The sample mean vector in group 1 is $\overline{\mathbf{X}}_1 = (86.38, 77.18, 80.5)^\mathsf{T}$ and in group 2 it is $\overline{\mathbf{X}}_2 = (90.62, 76.67, 88.57)^\mathsf{T}$. The sample covariances within each group and pooled are

$$S_1 = \begin{pmatrix} 151.2 & 93.9 & 157.7 \\ 93.9 & 259.5 & 86.8 \\ 157.7 & 86.8 & 731.5 \end{pmatrix},$$

$$S_2 = \begin{pmatrix} 114.5 & 30.4 & 9.4 \\ 30.4 & 203.3 & 62.5 \\ 9.4 & 62.5 & 152.9 \end{pmatrix},$$

$$S_p = \begin{pmatrix} 137.4 & 69.9 & 101.7 \\ 69.9 & 238.3 & 77.6 \\ 101.7 & 77.6 & 513.2 \end{pmatrix}.$$

Assuming $\Sigma_1 = \Sigma_2$, we get the test statistic

$$T^2 = \left(\frac{1}{n_1} + \frac{1}{n_2}\right)^{-1} (\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2)^\mathsf{T} S_p^{-1} (\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2) = 2.983,$$

so that $T^2(n_1 + n_2 - p - 1)/(p(n_1 + n_2 - 2)) = 0.9567$. Under the null hypothesis this statistic follows a $F_{p,n_1+n_2-p-1} = F_{3,51}$ distribution, so the P-value is equal to the tail area under this distribution, namely 0.4203.

If we do not assume $\Sigma_1 = \Sigma_2$, then we compute a different test statistic

$$T^2 = (\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2)^\mathsf{T} (S_1/n_1 + S_2/n_2)^{-1} (\overline{\mathbf{X}}_1 - \overline{\mathbf{X}}_2) = 3.764$$

which under the null hypothesis asymptotically follows a $\chi^2_p = \chi^2_3$ distribution. The P-value is the tail area under this distribution, namely 0.2881.

In this example, the $p$-value of null hypothesis that the two mean vectors are equal across $3^{rd}$ year and MSc students is quite large, no matter whether we assume $\Sigma_1 = \Sigma_2$ or not.

We remark that here we focused on computing test statistics and P-values, but we could equivalently have constructed a $1-\alpha$ confidence region for $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ and checked whether it contains the point $\mathbf{0}$.

### 5.3.4 Compare $K$ multivariate Normal means

We now consider comparing means across more than two groups, that is test the null hypothesis of constant mean vectors

$$H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \ldots = \boldsymbol{\mu}_K$$

where $K$ is the number of groups. This analysis is called Multivariate Analysis of Variance (MANOVA), and generalizes the univariate ANOVA as well as the Hotelling T-tests to compare means across more than 2 groups.

The MANOVA analysis is based on the following model for observation $i = 1, \ldots, n_k$ in group $k = 1, \ldots, K$,

$$\mathbf{X}_{ki} = \boldsymbol{\mu} + \boldsymbol{\delta}_k + \mathbf{e}_{ki},$$

where $\mathbf{e}_{ki} \sim N_p(\mathbf{0}, \Sigma)$ with independence across $k$ and $i$, and where $\boldsymbol{\mu} = \boldsymbol{\mu}_1$ and $\boldsymbol{\delta}_1 = \mathbf{0}$. Notice that the residuals can be correlated, but the covariance $\Sigma$ is assumed to be constant across all groups. The null hypothesis that all group means are equal can be re-stated as

$$H_0 : \boldsymbol{\delta}_2 = \ldots = \boldsymbol{\delta}_K = \mathbf{0}.$$

Similarly to a univariate ANOVA, we define within-groups and between-groups matrices (in the univariate case we have scalars rather than matrices). Let $\overline{\mathbf{X}}_k$ and $S_k$ be the mean and sample covariance in group $k$, and $\overline{\overline{\mathbf{X}}}$ the grand average across all groups. Then define

$$\begin{aligned}
\mathbf{W} &= \sum_{k=1}^{K} \sum_{i=1}^{n_k} (\mathbf{X}_{ki} - \overline{\mathbf{X}}_k)(\mathbf{X}_{ki} - \overline{\mathbf{X}}_k)^\mathsf{T} = \sum_{k=1}^{K} (n_k - 1) S_k, \\
\mathbf{B} &= \sum_{k=1}^{K} n_k (\overline{\mathbf{X}}_k - \overline{\overline{\mathbf{X}}})(\overline{\mathbf{X}}_k - \overline{\overline{\mathbf{X}}})^\mathsf{T},
\end{aligned}$$

where $\mathbf{W}$ is the within-groups sum of squares and cross-products matrix, and $\mathbf{B}$ is its between groups counterpart.

There are several possible tests based on the matrices $\mathbf{W}$ and $\mathbf{B}$, which become essentially equivalent as the sample size in each group grows. The most popular of these is probably Wilks' lambda test.

**Proposition 5.3.12 (not examinable).**
Assume that the MANOVA model holds and define Wilks' statistic $\Lambda^* = \frac{\det(\mathbf{W})}{\det(\mathbf{W}+\mathbf{B})}$. When $H_0 : \boldsymbol{\delta}_2 = \cdots = \boldsymbol{\delta}_K = \mathbf{0}$ holds and $n_k$ is large for all $k = 1, \ldots, K$, the test statistic

$$-\left(n - 1 - \frac{p+K}{2}\right) \log(\Lambda^*) \sim \chi^2_{p(K-1)}$$

approximately, where $n = \sum_{k=1}^{K} n_k$.

We skip the proof, as its quite convoluted, but note that the asymptotic distribution was proven by Bartlett. In some particular cases an exact distribution for $\Lambda^*$ is available, for instance when $p = 1$ and also when $K = 2$ or $K = 3$. These particular cases can be found in Johnson & Wichern's book. Furthermore, it can be shown that $\mathbf{W} + \mathbf{B} = \sum_k \sum_i (\mathbf{X}_{ki} - \overline{\overline{\mathbf{X}}})(\mathbf{X}_{ki} - \overline{\overline{\mathbf{X}}})^\mathsf{T}$. In R several versions of Wilks' lambda test are available in package `rrcov`, function `Wilks.test`.

**Example 5.3.13** (non-examinable)**.** Consider Anderson's Iris flower dataset, which records four variables for 150 plants. There are $n_1 = 50$ plants from the species *setosa*, $n_2 = 50$ from *versicolor* and $n_3 = 50$ from *virginica*. We wish to compare the vector with four means across the 3 groups, considering the null hypothesis

$$H_0 : \begin{pmatrix} \mu_{11} \\ \mu_{12} \\ \mu_{13} \\ \mu_{14} \end{pmatrix} = \begin{pmatrix} \mu_{21} \\ \mu_{22} \\ \mu_{23} \\ \mu_{24} \end{pmatrix} = \begin{pmatrix} \mu_{31} \\ \mu_{32} \\ \mu_{33} \\ \mu_{34} \end{pmatrix}.$$

The within and between groups sum-of-squares and cross-products matrices are

$$\mathbf{W} = \begin{pmatrix} 38.96 & 13.63 & 24.62 & 5.65 \\ 13.63 & 16.96 & 8.12 & 4.81 \\ 24.62 & 8.12 & 27.22 & 6.27 \\ 5.65 & 4.81 & 6.27 & 6.16 \end{pmatrix}, \tag{5.3.2}$$

$$\mathbf{B} = \begin{pmatrix} 63.21 & -19.95 & 165.25 & 71.28 \\ -19.95 & 11.34 & -57.24 & -22.93 \\ 165.25 & -57.24 & 437.10 & 186.77 \\ 71.28 & -22.93 & 186.77 & 80.41 \end{pmatrix},$$

with determinants $\det(\mathbf{W}) = 22096.88$ and $\det(\mathbf{W} + \mathbf{B}) = 942754.6$. We see that the determinant of the within-groups matrix is much smaller than that of the between-groups matrix, which suggests that there are differences between groups.

We compute the test statistic $-\left(n - 1 - \frac{p+K}{2}\right) \log(\Lambda^*) = 546.1$, which under a $\chi^2_\nu$ with $\nu = p(K-1) = 8$ degrees of freedom has a p-value essentially equal to 0 (up to numerical precision).

The R code required for the calculations is below. First the code to do step-by-step calculations is provided, and then the result is checked against that of function `Wilks.test`.

```
data(iris)

m1 <- colMeans(iris[iris$Species=='setosa',1:4])
m2 <- colMeans(iris[iris$Species=='versicolor',1:4])
m3 <- colMeans(iris[iris$Species=='virginica',1:4])
S1 <- cov(iris[iris$Species=='setosa',1:4])
S2 <- cov(iris[iris$Species=='versicolor',1:4])
S3 <- cov(iris[iris$Species=='virginica',1:4])
```

```r
n1 <- sum(iris$Species=='setosa')
n2 <- sum(iris$Species=='versicolor')
n3 <- sum(iris$Species=='virginica')
n <- n1+n2+n3; p <- 4; K <- 3

W <- (n1-1)*S1 + (n2-1)*S2 + (n3-1)*S3
m <- (n1*m1+n2*m2+n3*m3)/nrow(iris)
B <- n1*matrix(m1-m,ncol=1) %*% matrix(m1-m,nrow=1)
(n2*matrix(m2-m,ncol=1) %*% matrix(m2-m,nrow=1)) %>% round(2)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,]  0.43 -1.33  2.33  0.59
## [2,] -1.33  4.13 -7.21 -1.82
## [3,]  2.33 -7.21 12.60  3.18
## [4,]  0.59 -1.82  3.18  0.80
```

```r
(n3*matrix(m3-m,ncol=1) %*% matrix(m3-m,nrow=1)) %>% round(2)
```

```
##       [,1]  [,2]   [,3]  [,4]
## [1,] 27.73 -3.10  66.80 30.78
## [2,] -3.10  0.35  -7.48 -3.44
## [3,] 66.80 -7.48 160.92 74.15
## [4,] 30.78 -3.44  74.15 34.17
```

```r
det(W)
```

```
## [1] 22096.88
```

```r
det(W+B)
```

```
## [1] 457434.4
```

```r
l <- det(W)/det(W+B)
-(n-1-.5*(p+K)) * log(l)
```

```
## [1] 440.8937
```

```r
1-pchisq(-(n-1-.5*(p+K)) * log(l), df=p*(K-1))
```

```
## [1] 0
```

```r
#Same calculations, but using package rrcov
library(rrcov)
Wilks.test(x=iris[,1:4], grouping=iris$Species, method='c')
```

```
##
##  One-way MANOVA (Bartlett Chi2)
##
## data:  x
## Wilks' Lambda = 0.023439, Chi2-Value = 546.12, DF = 8.00, p-value < 2.2e-16
## sample estimates:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
```

---

### 5.3.5 Repeated measures analysis

Multivariate methods to compare mean vectors across groups are applied to a wide variety of problems. However, there is a specific setup where they are especially popular, which is referred to as *Repeated measures*.

Repeated measures studies collect multiple observations for each individual $i = 1, \ldots, n$. These observations are recorded at particular points in time $t = 1, \ldots, T$, for instance at one month ($t = 1$), three months ($t = 2$) and six months ($t = 3$) after surgery. In this situation, the observations for individual $i$ may be quite strongly correlated across $t = 1, \ldots, T$. Hence, we cannot assume independence and special data analysis methods are needed. Of course, there are other situations with correlated outcomes that can be tackled with a repeated measures analysis, but for simplicity we focus on studies with follow-up over time.

As an important particular case, when $T = 2$ we simply have two observations per subject. This is the standard setup for a paired t-test, in which one computes the differences between $t = 1$ and $t = 2$ (given by $y_{i2} - y_{i1}$) before performing any analysis. As we shall see, Repeated Measures Analysis is a generalization of the paired t-test when we have $T > 2$ measurements.

To fix ideas, let $\mathbf{X}_{ki} \sim N_T(\boldsymbol{\mu}_k, \Sigma)$ be the vector with the $T$ observations for individual $i = 1, \ldots, n_k$ in group $k = 1, \ldots, K$. A repeated measures analysis considers that the mean can change over time and also between groups, so we have a setup like that of a two-way ANOVA but where the residuals are expected to be correlated. As in a two-way ANOVA, we may be interested in testing for interactions between groups and time or in their main effects. As it turns out, these analyses can be performed in a straightforward manner using the multivariate tests we introduced in the previous sections.

Consider the example in Figure 43, showing the means for $K = 2$ groups followed at $T = 4$ time points. In the left panel groups do not interact with time, since the evolution over time is exactly the same in both groups. In statistical terms, the effects of group and time are additive. In the right panel we do observe an interaction between groups and time, as the grey group decreases faster than the

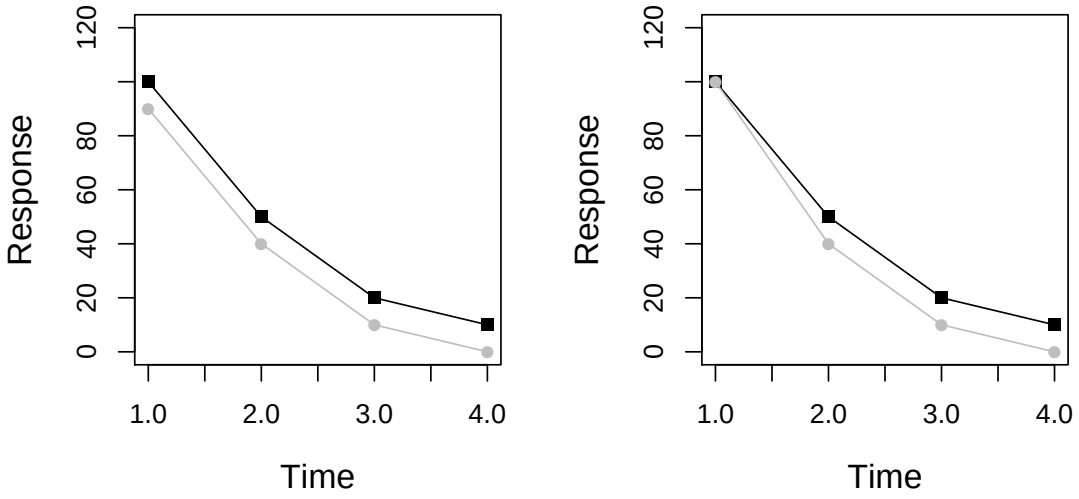



Figure 43: Repeated measures study with $T = 4$ time points and $K = 2$ groups. Left panel: no time and group interaction; Right panel: time and group interaction

black group. This corresponds to a situation where the effects of group and time are multiplicative.

When there is no interaction between group and time, so that each group evolves equally over time, one may wish to test whether there is any evolution over time at all. Statistically, this corresponds to testing for the main effect of time.

Similarly, in the absence of interactions one may wish to determine whether one group always presents higher mean values than the other. In statistical terms, this corresponds to testing for the main effect of the groups.

**Time and group interaction.**

Testing for interactions between time and group means that we want to determine whether the evolution over time is the same for all groups. That is, define evolution as the vector of mean differences over time in group $k$ as

$$\boldsymbol{\delta}_k = \begin{pmatrix} \mu_{k2} - \mu_{k1} \\ \mu_{k3} - \mu_{k2} \\ \cdots \\ \mu_{kT} - \mu_{kT-1} \end{pmatrix}.$$

In order to compare the evolution across groups the goal is to test

$$H_0 : \boldsymbol{\delta}_1 = \ldots = \boldsymbol{\delta}_K = \boldsymbol{\delta}.$$

Testing this hypothesis is straightforward by noting that $\boldsymbol{\delta}_k = C\boldsymbol{\mu}$, where

$$C = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & & \\ \cdots & & & & & \\ 0 & 0 & 0 & \ldots & -1 & 1 \end{pmatrix}. \tag{5.3.3}$$

Therefore, we can simply define $\mathbf{Y}_{ki} = C\mathbf{X}_{ki} \sim N_{T-1}(\boldsymbol{\delta}_k, C\Sigma C')$ and test $H_0$ either using Hotelling's $T^2$ statistic (for 2 groups) or Wilks' lambda statistic (for $> 2$ groups).

The hypothesis $H_0$ is equivalent to the existence of $\mu_0, (\lambda_k)_{k=2}^K$ and $(\nu_t)_{t=2}^T$ such that

$$\mu_{kt} = \mu_0 + \lambda_k + \nu_t,$$

where we write $\lambda_1 = \nu_1 = 0$. For this reason, a repeated measures model with no interaction is sometimes referred to as an *additive* model. This model is much simpler and easier to fit, having only $K + T - 1$ parameters compared with the $KT$ parameters of the full model.

**Effect of time.**

When the interaction between group and time is not statistically significant, it makes sense to consider if there is a main effect of time. That is, we want to test whether the means remain completely constant over time or not.

To perform this anaysis, we again define $\mathbf{Y}_{ki} = C\mathbf{X}_{ki}$ with $C$ as in (5.3.3). However, we now assume that there is no interaction between group and time, so that $\mathbf{Y}_{ki} \sim N_{T-1}(\boldsymbol{\delta}, C\Sigma C')$ where $\boldsymbol{\delta}$ now does not depend on $k$ but is common across all groups. The null hypothesis that there is no evolution over time is simply

$$H_0 : \boldsymbol{\delta} = \mathbf{0},$$

and hence the problem is reduced to a one-sample test for a multivariate Normal mean. As we saw in earlier sections, the test can be performed using Hotelling's $T^2$ statistic.

The null hypothesis that there is no interactive and no main effect of time is equivalent to the existence of $\mu_0$ and $(\lambda_k)_{k=2}^K$ such that

$$\mu_{kt} = \mu_0 + \lambda_k,$$

where we write $\lambda_1 = 0$.

**Main effect of groups.**

When there is no interaction between groups and time it makes sense to test for the main effects of the groups. That is, given that all groups evolve in a similar manner over time, we wish to compare their means. The corresponding null hypothesis is $H_0 : \boldsymbol{\mu}_1 = \cdots = \boldsymbol{\mu}_K$, which can be easily tested using either Hotelling's $T^2$ or Wilks' lambda statistic, as we saw in previous sections.

The null hypothesis that there is no interactive and no main effect of group is equivalent to the existence of $\mu_0$ and $(\nu_t)_{t=2}^T$ such that

$$\mu_{kt} = \mu_0 + \nu_t,$$

where we write $\nu_1 = 0$.

**Intercept-only**

If we find that there is no interaction, and moreover we find that there is no effect of time (or no effect of group, depending on which test we chose to do), we may wonder whether there is any difference between the data points at all. The simplest model that we may consider, the intercept-only model, says that neither group nor time has an effect, and we simply have $\mu_{kt} = \mu_0$ for all $k = 1, \ldots, K$ and $t = 1, \ldots, T$.

**Example 5.3.14.** We take the body fat dataset from R package `cccrm`, which measures the percentage of body fat on a series of girls at the ages of 12.5, 13 and 13.5 years. For each girl the three measurements are taken, so we have a repeated measures study. For 82 girls body fat was estimated from skinfold calipers (method 1) and for 82 more girls it was estimated with a method called DEXA (method 2).

```
library(cccrm)
data(bfat)
#colnames(bfat)[4] <- 'met'

visit11 <- bfat[bfat$VISITNO==2 & bfat$MET==1, ]
rownames(visit11) <- as.character(visit11$SUBJECT)
visit12 <- bfat[bfat$VISITNO==3 & bfat$MET==1, ]
rownames(visit12) <- as.character(visit12$SUBJECT)
visit13 <- bfat[bfat$VISITNO==4 & bfat$MET==1, ]
rownames(visit13) <- as.character(visit13$SUBJECT)
visit21 <- bfat[bfat$VISITNO==2 & bfat$MET==2, ]
rownames(visit21) <- as.character(visit21$SUBJECT)
visit22 <- bfat[bfat$VISITNO==3 & bfat$MET==2, ]
rownames(visit22) <- as.character(visit22$SUBJECT)
visit23 <- bfat[bfat$VISITNO==4 & bfat$MET==2, ]
rownames(visit23) <- as.character(visit23$SUBJECT)

obs1 <- data.frame(subject=visit11$SUBJECT,
                   bf1=visit11$BF,bf2=visit12$BF,bf3=visit13$BF)
obs2 <- data.frame(subject=visit21$SUBJECT,
                   bf1=visit21$BF,bf2=visit22$BF,bf3=visit23$BF)

m1 <- colMeans(obs1[,-1]); S1 <- cov(obs1[,-1]); n1 <- nrow(obs1)
m2 <- colMeans(obs2[,-1]); S2 <- cov(obs2[,-1]); n2 <- nrow(obs2)

err1 <- 1.96*sqrt(diag(S1)/n1)
err2 <- 1.96*sqrt(diag(S2)/n2)

plot(c(12.5,13,13.5), m1,type='l',ylim=c(20,26),xlab='Age (years)',
     ylab='% body fat',cex.lab=1.25)
points(c(12.5,13,13.5), m1,pch=15)
```

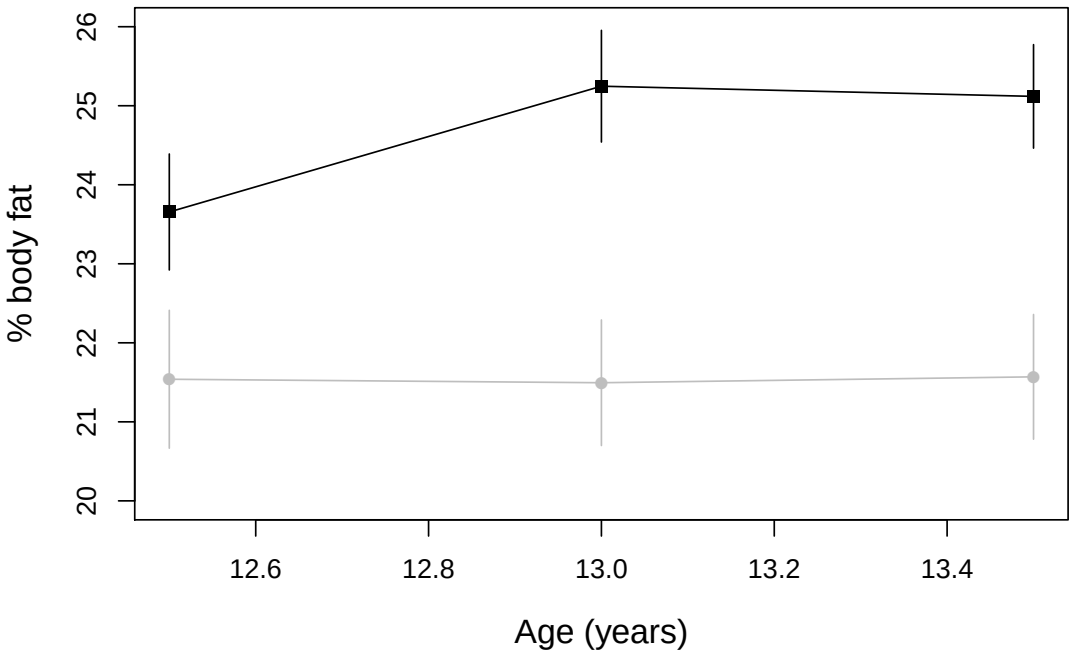



Figure 44: Average % body fat at 12.5, 13 and 13.5 years. Black line: method 1; Grey line: method 2

```
lines(c(12.5,13,13.5), m2, col='gray')
points(c(12.5,13,13.5), m2,pch=16,col='gray')
segments(x0=c(12.5,13,13.5),y0=m1-err1,y1=m1+err1)
segments(x0=c(12.5,13,13.5),y0=m2-err2,y1=m2+err2,col='gray')
```

Figure 44 shows the means and 95% univariate confidence intervals for both groups. We observe an increase between 12.5 and 13 years for method 1, whereas method 2 measurements seem to remain stable. We start the repeated measures analysis by testing for this hypothesized interaction between time and group. The null hypothesis is

$$H_0 : \begin{pmatrix} \mu_{12} - \mu_{11} \\ \mu_{13} - \mu_{12} \end{pmatrix} = \begin{pmatrix} \mu_{22} - \mu_{21} \\ \mu_{23} - \mu_{22} \end{pmatrix}.$$

To test $H_0$ we compute the differences between successive visits, *i.e.* define $\mathbf{Y} = (X_2 - X_1, X_3 - X_2)'$. The mean and covariance of $\mathbf{Y}$ in each group are

$$\overline{\mathbf{Y}}_1 = \begin{pmatrix} 1.59 \\ -0.13 \end{pmatrix}; \quad S_1 = \begin{pmatrix} 3.01 & -0.57 \\ -0.57 & 3.06 \end{pmatrix}$$
$$\overline{\mathbf{Y}}_2 = \begin{pmatrix} -0.04 \\ 0.07 \end{pmatrix}; \quad S_2 = \begin{pmatrix} 2.23 & -1.10 \\ -1.10 & 3.51 \end{pmatrix}$$

Since the covariances do not appear to be extremely different, we assume $\Sigma_1 = \Sigma_2$

and compute the pooled covariance matrix

$$S_p = \begin{pmatrix} 2.62 & -0.84 \\ -0.84 & 3.28 \end{pmatrix}.$$

```
#Time * group interaction
C <- matrix(c(-1,1,0,0,-1,1),byrow=TRUE,nrow=2)
d1 <- as.matrix(obs1[,-1]) %*% t(C)
md1 <- C %*% matrix(m1,ncol=1); Sd1 <- C %*% S1 %*% t(C)
d2 <- as.matrix(obs2[,-1]) %*% t(C)
md2 <- C %*% matrix(m2,ncol=1); Sd2 <- C %*% S2 %*% t(C)
Sp <- ((n1-1)*Sd1 + (n2-1)*Sd2)/(n1+n2-2)
p <- 2

T2 <- matrix(md1-md2,nrow=1) %*% solve(Sp/n1 + Sp/n2) %*%
    matrix(md1-md2,ncol=1)
f <- (n1+n2-p-1) / (p * (n1+n2-2)) * T2
pval <- 1 - pf(f, df1=p,df2=n1+n2-p-1)
T2

##          [,1]
## [1,] 43.22403

f

##          [,1]
## [1,] 21.47861

pval

##              [,1]
## [1,] 5.389975e-09
```

Hotelling's statistic is therefore

$$T^2 = (\overline{\mathbf{Y}}_1 - \overline{\mathbf{Y}}_2)^\mathsf{T} S_p^{-1} (\overline{\mathbf{Y}}_1 - \overline{\mathbf{Y}}_2)(1/n_1 + 1/n_2)^{-1} = 43.22,$$

so that

$$\frac{(n_1 + n_2 - 2 - 1)}{2(n_1 + n_2 - 2)} T^2 = 21.479,$$

and its corresponding p-value under an $F_{2,n_1+n_2-2-1}$ is $5.39 \times 10^{-9}$. There is a highly significant interaction between time and the measurement method, matching our observation in Figure 44.

```
#Effect of time
T2 <- n1 * matrix(md1,nrow=1) %*% solve(Sd1) %*% matrix(md1,ncol=1)
f <- T2 * (n1-p)/((n1-1)*p)
```

```
pval <- 1 - pf(f, df1=p,df2=n1-p)
T2

##           [,1]
## [1,] 69.75136

f

##           [,1]
## [1,] 34.44512

pval

##               [,1]
## [1,] 1.617906e-11

T2 <- n2 * matrix(md2,nrow=1) %*% solve(Sd2) %*% matrix(md2,ncol=1)
f <- T2 * (n2-p)/((n2-1)*p)
pval <- 1 - pf(f, df1=p,df2=n2-p)
T2

##           [,1]
## [1,] 0.1484117

f

##            [,1]
## [1,] 0.07328975

pval

##           [,1]
## [1,] 0.9293938
```

Now that we have seen that there is an interaction, we might be interested in testing whether there are any differences over time within each group. To test the effect of time within group 1 we compute $T^2 = n_1 \overline{\mathbf{Y}}' S_1^{-1} \overline{\mathbf{Y}} = 69.751$, so that

$$\frac{n_1 - 2}{(n_1 - 1)2} T^2 = 34.445$$

and the corresponding P-value under an $F_{2,n_1-p}$ is $1.62 \times 10^{-11}$. To test the effect of time within group 2 we find $T^2 = n_2 \overline{\mathbf{Y}}' S_2^{-1} \overline{\mathbf{Y}} = 0.148$, so that

$$\frac{n_2 - 2}{(n_2 - 1)2} T^2 = 0.073$$

and the corresponding P-value under an $F_{2,n_2-p}$ is 0.9294. Therefore, we find statistically significant changes over time for method 1 but not for method 2, again

matching what we observe in Figure 44.

## 5.4 Checking multivariate Normality

Most of the results we saw either assume a multivariate Normal distribution or that the sample size is large enough for the Central Limit Theorem to kick in. Of course, in real life data are never truly Normally distributed, but when the sample size is moderately small it is useful to assess whether the Normal distribution is a reasonable approximation. There are various methods for checking multivariate normality, and a good R package implementing them is `MVN`. We will only discuss some simple tests for assessing multivariate normality.

First, recall that if $\mathbf{X}_1, \ldots, \mathbf{X}_n \overset{\text{iid}}{\sim} \mathbf{X}$ are i.i.d. MVN, then any 1D linear projections of them are MVN (Proposition 5.1.6). In fact, this is even a characterization of the MVN distribution. So now we know that for any non-random $\mathbf{v} \in \mathbb{R}^p$, $\mathbf{v}^\mathsf{T}\mathbf{X}_1, \ldots, \mathbf{v}^\mathsf{T}\mathbf{X}_n$ should be an i.i.d. sample from a univariate normal distribution. Therefore one can pick a couple of univariate projections (given by $\mathbf{v}$s), and check if they are Gaussian. This can be done for instance by comparing histograms with the Gaussian density, or by producing a Gaussian QQ-plot.

**Example 5.4.1.** Consider the dataset rating the importance of $p = 11$ items for $n = 63$ students to be happy with a course. In order to assess the extent to which the multivariate Normal is a reasonable assumption, we first start by looking at the distribution of each variable separately. We see a histogram of the first two variables in Figure 45. Although variable 2 could be Gaussian, variable 1 is highly non-Gaussian (there is no central mode), and therefore the data is not MVN.

```
## Check normality
x <- read.table('~/st323/data/course_happy_nomiss.txt',header=TRUE)
x <- as.matrix(x[,1:11])
m <- colMeans(x); S <- cov(x)
n <- nrow(x); p <- ncol(x)
d <- mahalanobis(x, center=m, cov=S) * (n-p)/((n-1)*p)


op <- par(mfrow=c(1,2))
hist(x[,1], 'FD')
hist(x[,2], 'FD')
par(op)
```

Using the `MVN` package, we get the following output:

```
library(MVN)

## sROC 0.1-2 loaded

mvn_test <- mvn(x)
mvn_test$multivariateNormality
```

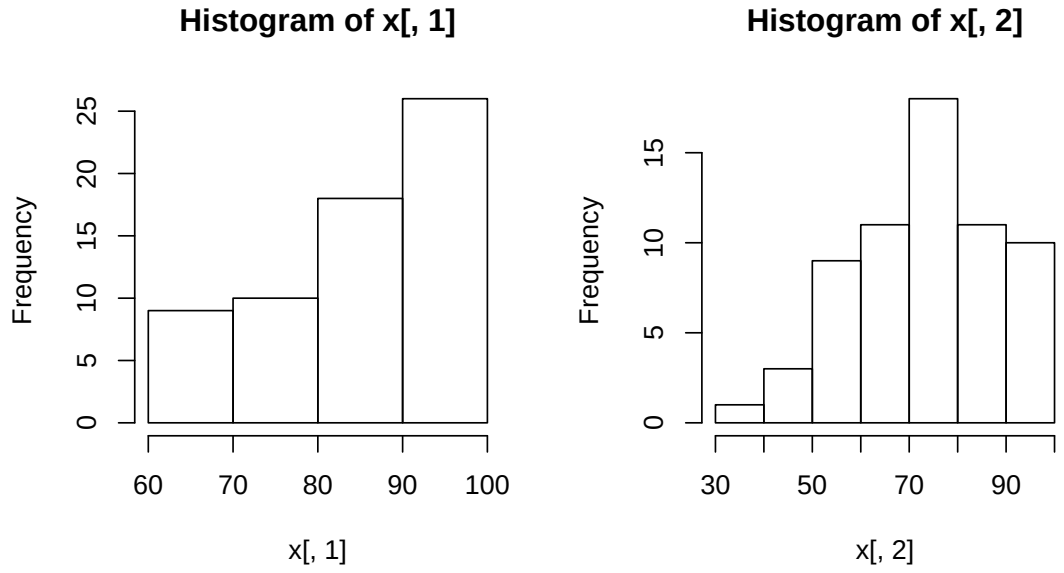



Figure 45: Histogram of the first variable (left) and the second variable (right) for the "happy with the course" dataset.

```
##              Test        Statistic              p value Result
## 1 Mardia Skewness 570.960904545477 3.75421755283091e-21     NO
## 2 Mardia Kurtosis 9.35237199555605                    0     NO
## 3             MVN             <NA>                 <NA>     NO

mvn_test$univariateNormality

##            Test                      Variable Statistic   p value Normality
## 1  Shapiro-Wilk      Teacher_explaining         0.8720  <0.001      NO
## 2  Shapiro-Wilk     Teacher_enthusiastic        0.9531  0.0176      NO
## 3  Shapiro-Wilk Teacher_material_interesting    0.8559  <0.001      NO
## 4  Shapiro-Wilk     Material_stimulating        0.9222   7e-04      NO
## 5  Shapiro-Wilk       Material_useful           0.8724  <0.001      NO
## 6  Shapiro-Wilk        Clear_marking            0.9024   1e-04      NO
## 7  Shapiro-Wilk         Marking_fair            0.8360  <0.001      NO
## 8  Shapiro-Wilk       Feedback_prompt           0.9312  0.0017      NO
## 9  Shapiro-Wilk      Feedback_clarifies         0.8178  <0.001      NO
## 10 Shapiro-Wilk      Detailed_comments          0.8709  <0.001      NO
## 11 Shapiro-Wilk            Notes                0.7340  <0.001      NO
```

In particular, the "Mardia Skewness" test (which tests joint normality of the variables) gives a ridiculously small p-value (under the null hypothesis that the data is MVN). This corroborates our conclusion that the data is not MVN. The variable-wise tests "Shapiro-Wilk" also corroborate that the first variable is non-Gaussian, but provide in comparison weaker evidence that the second variable is non-Gaussian. Notice that the package does not returns the true p-value for the test on the first

variable, but only returns "$< 0.001$". This is bad practice, and should be avoided.

---

# 6 Classification (or Supervised Learning)

## 6.1 Basic Theory of Classification

The goal in classification is to assign individuals to a group (or class) amongst several possible groups. Here we *do know* which groups are present in the data (in contrast to clustering, which we shall see later), and we want to classify the data into these groups. For clarity here we focus on the case with two groups, but most of the ideas we will see extend directly to cases with an arbitrary number of groups.

Let $Y = 1$ indicate that an individual belongs to group 1, and $Y = 2$ that it belongs to group 2. Also suppose that we observe a set of variables $\mathbf{X}$ for that individual. The goal is to guess the group given that we observe $\mathbf{X} = \mathbf{x}$. Here we will assume that $\mathbf{x} \in \mathbb{R}^p$, although in general it could be any kind of data that can help us predict $Y$. Of course, the underlying hope is that $\mathbf{X}$ has some predictive power that will help us discern the groups, otherwise our predictions will be no better than random guesses.

Denote by $f(\mathbf{x} \mid Y = 1)$ the probability density function of $\mathbf{X}$ for individuals in group 1, and $f(\mathbf{x} \mid Y = 2)$ that for group 2. Further, let $\pi_1 = \mathbb{P}(Y = 1)$ and $\pi_2 = \mathbb{P}(Y = 2)$ be the prior probabilities for each class before observing $\mathbf{x}$. From these we can find the probability that an individual with variables $\mathbf{x}$ belongs to each group, namely

$$\mathbb{P}(Y = 1 \mid \mathbf{x}) = \frac{f(\mathbf{x} \mid Y = 1)\pi_1}{f(\mathbf{x} \mid Y = 1)\pi_1 + f(\mathbf{x} \mid Y = 2)\pi_2} \tag{6.1.1}$$

and $\mathbb{P}(Y = 2 \mid \mathbf{x}) = 1 - \mathbb{P}(Y = 1 \mid \mathbf{x})$. The result follows directly from Bayes' theorem. Notice that when $X$ and $Y$ are independent, then $f(\mathbf{x} \mid Y = 1) = f(\mathbf{x} \mid Y = 2)$ and hence $\mathbb{P}(Y = 1 \mid \mathbf{x}) = \pi_1$.

We now define the concept of a *classification rule*.

**Definition 6.1.1 (Classification rule).**
A classification rule is a function mapping $\mathbf{x} \in \mathbb{R}^p$ to $\{1, 2\}$. Equivalently, define the classification regions $R_1$ and $R_2 = \mathbb{R}^p \setminus R_1$ such that

- If $\mathbf{x} \in R_1$, we predict $Y = 1$,
- If $\mathbf{x} \in R_2$, we predict $Y = 2$.

Figure 46 shows an example with a classification rule for $\mathbf{x} \in \mathbb{R}^2$. According to this rule, all observations falling on the left of the curvy boundary are assigned to group 1, whereas observations to the right of the boundary are allocated to group 2.

Our goal will be to define classification rules that perform well in terms of correctly classifying individuals or, more generally, minimizing the cost of misclassifications. Given any classification rule, the probabilities of correctly classifying an

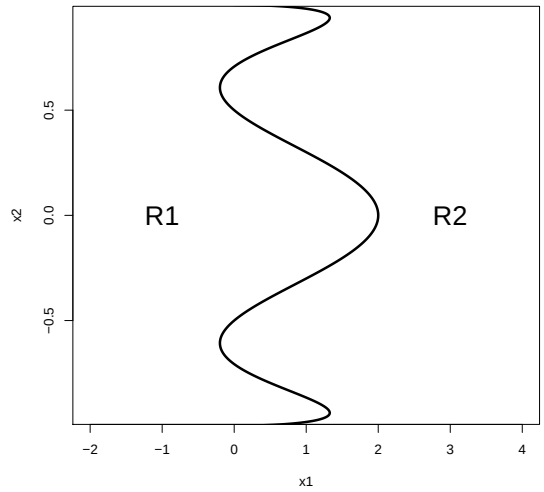



Figure 46: Classification rule for $\mathbf{x} \in \mathbb{R}^2$

individual into each class are

$$\mathbb{P}(Y = 1, \mathbf{x} \in R_1) = \mathbb{P}(\mathbf{x} \in R_1 \mid Y = 1)\pi_1 = \pi_1 \int_{\mathbf{x} \in R_1} f(\mathbf{x} \mid Y = 1) d\mathbf{x}$$

and

$$\mathbb{P}(Y = 2, \mathbf{x} \in R_2) = \mathbb{P}(\mathbf{x} \in R_2 \mid Y = 2)\pi_2 = \pi_2 \int_{\mathbf{x} \in R_2} f(\mathbf{x} \mid Y = 2) d\mathbf{x}.$$

Similarly, the misclassification probabilities are

$$\mathbb{P}(Y = 2, \mathbf{x} \in R_1) = \mathbb{P}(\mathbf{x} \in R_1 \mid Y = 2)\pi_2 = \pi_2 \int_{\mathbf{x} \in R_1} f(\mathbf{x} \mid Y = 2) d\mathbf{x}$$

and

$$\mathbb{P}(Y = 1, \mathbf{x} \in R_2) = \mathbb{P}(\mathbf{x} \in R_2 \mid Y = 1)\pi_1 = \pi_1 \int_{\mathbf{x} \in R_2} f(\mathbf{x} \mid Y = 1) d\mathbf{x}.$$

A common goal in a classification analysis is to achieve high correct classification and low misclassification probabilities. However, one should also consider that the consequences of each correct and incorrect decision can be quite different. For instance, scientists in Italy recently found that failing to predict an earthquake has much more severe consequences (for everyone involved) than incorrectly predicting that there will be an earthquake. In other words, each misclassification can have different costs, which can be easily taken into account by considering the following table

| | Truth | |
|---|---|---|
| Prediction | $Y = 1$ | $Y = 2$ |
| $g(\mathbf{x}) = 1$ | 0 | $c_2$ |
| $g(\mathbf{x}) = 2$ | $c_1$ | 0 |

where $g : \mathbb{R}^p \to \{1, 2\}$ is a classification rule, and $c_2, c_1 \in \mathbb{R}^+$. That is, $c_2$ is the cost of misclassifying an observation (to group 1) when it truly belongs to group 2, and $c_1$ of misclassifying (to group 2) an observation that truly belongs to group 1. As we shall see in a moment, only the ratio $c_2/c_1$ matters so without loss of generality we can set $c_1 = 1$. Following a decision-theoretic argument, a reasonable goal is to minimize the *expected cost of misclassification*. Let $\text{cost}(Y, g(X))$ be the function defined in the table above.

**Proposition 6.1.2 (Expected cost of misclassification).**
Suppose we use the classification rule $g : \mathbb{R}^p \to \{1, 2\}$, that assigns to group 1 when $\mathbf{x} \in R_1$ and to group 2 when $\mathbf{x} \in R_2$. The expected cost of misclassification associated to the rule $g$ is

$$\mathbb{E}\left[\text{cost}(Y, g(X))\right] = c_2\, \mathbb{P}(\mathbf{x} \in R_1 \mid Y = 2)\pi_2 + c_1\, \mathbb{P}(\mathbf{x} \in R_2 \mid Y = 1)\pi_1 \tag{6.1.2}$$

*Proof.* Left as a exercise. □

The following result tells us that one can minimize the expected misclassification cost using a fairly simple classification rule.

**Proposition 6.1.3 (Bayes' Classifier).**
The rule minimizing the expected cost of misclassification (6.1.2) for a given $\mathbf{x}$ is to define $R_1$ as

$$R_1 \triangleq \{\mathbf{x} : f(\mathbf{x} \mid Y = 1)\pi_1 c_1 > f(\mathbf{x} \mid Y = 2)\pi_2 c_2\},$$

and $R_2$ as its complement. Equivalently,

$$R_1 \triangleq \left\{\mathbf{x} : \frac{f(\mathbf{x} \mid Y = 1)\pi_1}{f(\mathbf{x} \mid Y = 2)\pi_2} > \frac{c_2}{c_1}\right\}.$$

This classification rule is called the **Bayes classifier**.

*Proof.* See video.

□

As an important remark, the optimal rule requires knowing $f(\mathbf{x} \mid Y = 1)$ and $f(\mathbf{x} \mid Y = 2)$. In general these densities are not known and must be estimated from the data, in which case Proposition 6.1.3 does not directly apply anymore. The Proposition also requires the class prior probabilities $\pi_1$,$\pi_2$, which may not always be available.

**Remark 6.1.4 (Bayes' classifier using posterior class probabilities).** Assuming that the marginal distribution of $\mathbf{X} \in \mathbb{R}^p$ has a density $f$, notice that we only need to define the classification rule on the set $\{\mathbf{x} \in \mathbb{R}^p : f(x) > 0\}$. In this case, we have that Bayes' classifier can be

re-written as

$$R_1 \triangleq \{\mathbf{x} \in \mathbb{R}^p : c_1\, \mathbb{P}(Y = 1|\mathbf{X} = \mathbf{x}) > c_2\, \mathbb{P}(Y = 2|\mathbf{X} = \mathbf{x})\}.$$

It turns out it that is possible to extend Proposition 6.1.3 within the Bayesian paradigm and find the optimal rule when the densities are not known, but we shall not pursue this here. We will simply note that in practice one typically estimates the required quantities, as we shall see later on.

Proposition 6.1.3 tells us a general recipe to obtain the optimal rule. The expression can be simplified even further when either the costs or the prior probabilities are equal.

**Proposition 6.1.5 (Bayes' classifier in special cases).**

1. Equal prior probabilities. If $\pi_1 = \pi_2$ then

$$R_1 \triangleq \{\mathbf{x} : f(\mathbf{x} \mid Y = 1)c_1 > f(\mathbf{x} \mid Y = 2)c_2\}\,.$$

2. Equal misclassification costs. If $c_2/c_1 = 1$ then

$$R_1 \triangleq \{\mathbf{x} : f(\mathbf{x} \mid Y = 1)\pi_1 > f(\mathbf{x} \mid Y = 2)\pi_2\}\,.$$

3. Equal prior probabilities and costs. If $\pi_1 = \pi_2$ and $c_2/c_1 = 1$ then

$$R_1 \triangleq \{\mathbf{x} : f(\mathbf{x} \mid Y = 1) > f(\mathbf{x} \mid Y = 2)\}\,.$$

In fact, many classification procedures attempt to classify individuals into the class with highest probability. Such methods make the implicit assumption that $c_2 = c_1$, i.e. equal misclassification costs.

There are also methods that classify individuals using only the likelihood under each class $f(\mathbf{x} \mid Y = 1)$ and $f(\mathbf{x} \mid Y = 2)$. Even if $c_2 = c_1$, these methods will be sub-optimal unless $\pi_1 = \pi_2$. That is, whenever possible we want to take into account the prior probability of each class.

**Example 6.1.6.** Suppose that physicists observe certain measurements $\mathbf{x}$ that help predict whether an earthquake will take place ($Y = 1$) or not ($Y = 2$) in the next 24 hours. Further suppose that based on historical information, it is known that the density of $\mathbf{x}$ when there truly is going to be an earthquate is $f(\mathbf{x} \mid Y = 1) = 0.25$. Also, it is known that $f(\mathbf{x} \mid Y = 2) = 0.01$. That is, the observed data are much more likely when there is going to be an earthquake.

Suppose that from historical data we also know that on average only 1 earthquake per year occurs in the location of interest. That is, $\pi_1 = 1/365$ and $\pi_2 = 1 - \pi_1$. After observing $\mathbf{x}$, the probability of an earthquake in the next 24 hours is

$$\mathbb{P}(Y = 1 \mid \mathbf{x}) = \frac{f(\mathbf{x} \mid Y = 1)\pi_1}{f(\mathbf{x} \mid Y = 1)\pi_1 + f(\mathbf{x} \mid Y = 2)\pi_2}$$

$$= \frac{0.25\frac{1}{365}}{0.25\frac{1}{365} + 0.01\frac{364}{365}} = 0.0643.$$

Based on the observed data, there is a fairly high probability that the earthquake will not happen. However, a prudent decision maker should also consider the cost of making a wrong decision. Because failing to predict an actual earthquake has very severe consequences, an expert tells us that $c_2 = 1$ and $c_1 = 100$. We therefore get

$$c_1 \, \mathbb{P}(Y = 1 \mid \mathbf{x}) = 6.43$$

and

$$c_2 \, \mathbb{P}(Y = 2 \mid \mathbf{x}) = 0.9357.$$

Based on these Bayes' classifier tells us to predict that an earthquake is going to happen ($Y = 1$), despite its actual probability being fairly low.

As an exercise, let us determine the range of costs $c_1$ for which the optimal decision would still be to predict an earthquake (we set $c_2 = 1$ without loss of generality). We should predict $Y = 1$ whenever

$$\begin{aligned} & f(\mathbf{x} \mid Y = 1)\pi_1 c_1 > f(\mathbf{x} \mid Y = 2)\pi_2 \\ \iff \quad & 0.25\frac{1}{365}c_1 > 0.01\frac{364}{365} \\ \iff \quad & c_1 > 14.56. \end{aligned}$$

That is, as long as the cost of failing to predict an earthquake is at least 14.56 times that of predicting an earthquake that actually fails to occur, we should predict an earthquake to avoid potential catastrophic consequences.

---

## 6.2 Classification for multivariate Normal predictors

Recall that Proposition 6.1.3 gives us the general recipe to obtain optimal classifications for any class-specific densities $f(\mathbf{x} \mid Y = 1)$ and $f(\mathbf{x} \mid Y = 2)$. It is interesting to study the case where both densities are Normal, as then the resulting rule has an intuitive, appealing interpretation. Specifically, in this Section we assume that $X|Y = y \sim N_p(\mu_y, \Sigma_y)$ for $y = 1, 2$, where both $\Sigma_1$ and $\Sigma_2$ are SPD.

Consider first the case with equal covariances $\Sigma_1 = \Sigma_2$. We have the following result:

**Proposition 6.2.1 (Optimal rule for MVN data and equal covariances).**

Let $f(\mathbf{x} \mid Y = y)$ be the density of a $N_p(\boldsymbol{\mu}_y, \Sigma)$, $y = 1, 2$. Letting $\mathbf{b} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$, the Bayes classifier assigns a new observation $\mathbf{x} \in \mathbb{R}^p$ to group 1 when

$$\mathbf{x}^\mathsf{T}\mathbf{b} > \frac{1}{2}(\boldsymbol{\mu}_2 + \boldsymbol{\mu}_1)^\mathsf{T}\mathbf{b} + \log\left(\frac{\pi_2}{\pi_1}\frac{c_2}{c_1}\right)$$

and to group 2 otherwise.

*Proof.* See video. The ratio of likelihoods in Proposition 6.1.3 becomes

$$\begin{aligned}\frac{f(\mathbf{x} \mid Y=1)}{f(\mathbf{x} \mid Y=2)} &= e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)} \\ &= e^{\frac{1}{2}\boldsymbol{\mu}_2^\mathsf{T}\Sigma^{-1}\boldsymbol{\mu}_2-\frac{1}{2}\boldsymbol{\mu}_1^\mathsf{T}\Sigma^{-1}\boldsymbol{\mu}_1-\mathbf{x}^\mathsf{T}\Sigma^{-1}\boldsymbol{\mu}_2+\mathbf{x}^\mathsf{T}\Sigma^{-1}\boldsymbol{\mu}_1} \\ &= e^{\frac{1}{2}\left(\boldsymbol{\mu}_2^\mathsf{T}\Sigma^{-1}\boldsymbol{\mu}_2-\boldsymbol{\mu}_1^\mathsf{T}\Sigma^{-1}\boldsymbol{\mu}_1\right)+\mathbf{x}^\mathsf{T}\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)} \\ &= e^{\frac{1}{2}(\boldsymbol{\mu}_2+\boldsymbol{\mu}_1)^\mathsf{T}\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)+\mathbf{x}^\mathsf{T}\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)}.\end{aligned}$$

Therefore, the optimal rule is to classify into group 1 whenever

$$\frac{1}{2}(\boldsymbol{\mu}_2+\boldsymbol{\mu}_1)^\mathsf{T}\Sigma^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)+\mathbf{x}^\mathsf{T}\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) > \log\left(\frac{\pi_2}{\pi_1}\frac{c_2}{c_1}\right).$$

□

Recall that $\mathbf{x}^\mathsf{T}\mathbf{b}$ is the projection of $\mathbf{x}$ on $\mathbf{b}$. The rule compares the projection of $\mathbf{x}$ with that of the midpoint between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$. In particular, when $\pi_1 = \pi_2$ and $c_2 = c_1$ the rule is to assign to group 1 whenever the projection of $\mathbf{x}$ is closer to $\boldsymbol{\mu}_1$ than to $\boldsymbol{\mu}_2$, and vice versa. The expression also reveals that the optimal rule is linear in $\mathbf{x}$.

**Example 6.2.2.** Consider a case of two Normal populations with

$$\boldsymbol{\mu}_1 = \begin{pmatrix}0\\0\end{pmatrix}; \boldsymbol{\mu}_2 = \begin{pmatrix}3\\3\end{pmatrix}; \Sigma_1 = \Sigma_2 = \begin{pmatrix}1 & -0.5\\ -0.5 & 1\end{pmatrix}.$$

Figure 47 (top subfigures) shows the contours for the densities in each group $f(\mathbf{x} \mid Y=1)$ and $f(\mathbf{x} \mid Y=2)$. Let us first assume that $\pi_1 = \pi_2$ and $c_2 = c_1$ and let $\mathbf{b} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = (-6,-6)^\mathsf{T}$, then the optimal decision rule is to assign to group 1 whenever

$$\mathbf{x}^\mathsf{T}\begin{pmatrix}-6\\-6\end{pmatrix} > (1.5, 1.5)\begin{pmatrix}-6\\-6\end{pmatrix} = -18 \iff -6x_1 - 6x_2 > -18 \iff x_2 < 3 - x_1,$$

which is a linear decision rule. The optimal boundary is shown in the top left panel of Figure 47 as a straight line.

Suppose now that group 1 has higher *a priori* probability than group 2, say $\pi_1 = 0.9$ and $\pi_2 = 0.1$. These prior probabilities might arise, for instance, because group 1 corresponds to healthy individuals and group 2 to a relatively uncommon disease. The new optimal rule is given by

$$-6x_1 - 6x_2 > -18 + \log(1/9) \iff x_2 < 3.366 - x_1.$$

That is, we obtain a line with the same slope but different intercept, shown in the top right panel of Figure 47. Here the effect of unequal prior probabilities becomes

obvious. Because group 2 is much less common (a priori) than group 1, we allocate individuals to group 1 unless there is overwhelming evidence in favor of group 2 (in the sense that the density is much higher).

Figure 47 (bottom subfigures) shows the optimal classification rule for the same $\boldsymbol{\mu}_1, \Sigma$ as above, but for $\boldsymbol{\mu}_2 = (-1, 4)^\mathsf{T}$. The bottom left subfigure is for $\pi_1 = \pi_2, c_1 = c_2$, and the bottom right subfigure is for $\pi_1 = 0.9, c_1 = c_2$. Notice that the classification boundary line is not aligned with any eigenvector of the covariance matrix $\Sigma$.

```
## Optimal rule for Normal case, Sigma1=Sigma2

m2list <- list(c(3,3), c(-1,4))

op <- par(mfrow=c(2,2), mai=rep(.1, 4))

for(i in seq(along=m2list)){
m1 <- c(0,0);
m2 <- m2list[[i]]
S1 <- S2 <- matrix(c(1,-.5,-.5,1),nrow=2)
m <- .5*(m1+m2); b <- as.vector(solve(S1) %*% matrix(m1-m2,ncol=1))
mb <- sum(m*b)
pi1 <- 0.9; pi2 <- 0.1


library(mvtnorm)

xseq <- seq(-3,3,length=100)
xgrid <- expand.grid(xseq,xseq)
fx <- dmvnorm(xgrid, mean=m1, sigma=S1)
contour(x=xseq,y=xseq,z=matrix(fx,nrow=length(xseq)), xlim=c(-3,6),
        ylim=c(-3,6),drawlabels=FALSE, xaxt='n', yaxt='n')
xseq <- seq(-6,8,length=100)
xgrid <- expand.grid(xseq,xseq)
fx <- dmvnorm(xgrid, mean=m2, sigma=S2)
#par(new=TRUE)
contour(x=xseq,y=xseq,z=matrix(fx,nrow=length(xseq)), xlim=c(-3,6),
        ylim=c(-3,6),drawlabels=FALSE,lwd=1,lty=2, add=TRUE)
#
abline(mb/b[2], -b[1]/b[2], col='blue', lwd=3)
text(-2,-2,'R1',cex=2); text(5,5,'R2',cex=2)


xseq <- seq(-3,3,length=100)
xgrid <- expand.grid(xseq,xseq)
fx <- dmvnorm(xgrid, mean=m1, sigma=S1)
contour(x=xseq,y=xseq,z=matrix(fx,nrow=length(xseq)), xlim=c(-3,6),
```

```
        ylim=c(-3,6),drawlabels=FALSE, xaxt='n', yaxt='n')
xseq <- seq(-6,8,length=100)
xgrid <- expand.grid(xseq,xseq)
fx <- dmvnorm(xgrid, mean=m2, sigma=S2)
contour(x=xseq,y=xseq,z=matrix(fx,nrow=length(xseq)), xlim=c(-3,6),
        ylim=c(-3,6),drawlabels=FALSE,lwd=1,lty=2, add=TRUE)
#
abline(mb/b[2]-log(pi2/pi1), -b[1]/b[2], col='blue', lwd=3)
text(-2,-2,'R1',cex=2); text(5,5,'R2',cex=2)
}
par(op)
```

---

Let us now consider the case where covariances are different. We have the following result.

**Proposition 6.2.3 (Optimal rule for Normal data and unequal covariances).**
Let $f(\mathbf{x} \mid Y = y)$ be the density of a $N_p(\boldsymbol{\mu}_y, \Sigma_y)$, for $y = 1, 2$. Bayes' classifier assign a new observation $\mathbf{x} \in \mathbb{R}^p$ to group 1 when

$$-\frac{1}{2}\mathbf{x}^\mathsf{T} B\mathbf{x} + \mathbf{x}^\mathsf{T}\mathbf{b} > \log\left(\frac{\pi_2}{\pi_1}\frac{c}{c_1}\right) - k$$

and to group 2 otherwise, where $B = \Sigma_1^{-1} - \Sigma_2^{-1}$, $\mathbf{b} = \Sigma_1^{-1}\boldsymbol{\mu}_1 - \Sigma_2^{-1}\boldsymbol{\mu}_2$ and

$$k = \frac{1}{2}\left(\log\left(\frac{\det(\Sigma_2)}{\det(\Sigma_1)}\right) + \boldsymbol{\mu}_2^\mathsf{T}\Sigma_2^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^\mathsf{T}\Sigma_1^{-1}\boldsymbol{\mu}_1\right).$$

*Proof.* This is similar to the proof of Proposition 6.2.1, but extended to the case where $\Sigma_1 \neq \Sigma_2$. Left as an exercise. □

The optimal rule when $\Sigma_1 \neq \Sigma_2$ depends on a quadratic function of $\mathbf{x}$. Of course, when $\Sigma_1 = \Sigma_2$ the expression simplifies to the linear rule given in Proposition 6.2.1.

**Example 6.2.4.** Consider a case of two Normal populations with

$$\boldsymbol{\mu}_1 = \begin{pmatrix}0\\0\end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix}1\\3\end{pmatrix}, \quad \Sigma_1 = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}, \quad \Sigma_2 = \begin{pmatrix}1 & 0.5\\0.5 & 1\end{pmatrix},$$

and consider equal prior class probabilities $\pi_1 = \pi_2$ and misclassification costs $c_2 = c_1$.

Doing the appropriate computations we obtain that $k = 4.52$ and

$$B = \begin{pmatrix}-\frac{1}{3} & \frac{2}{3}\\ \frac{2}{3} & -\frac{1}{3}\end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix}2/3\\-10/3\end{pmatrix}.$$

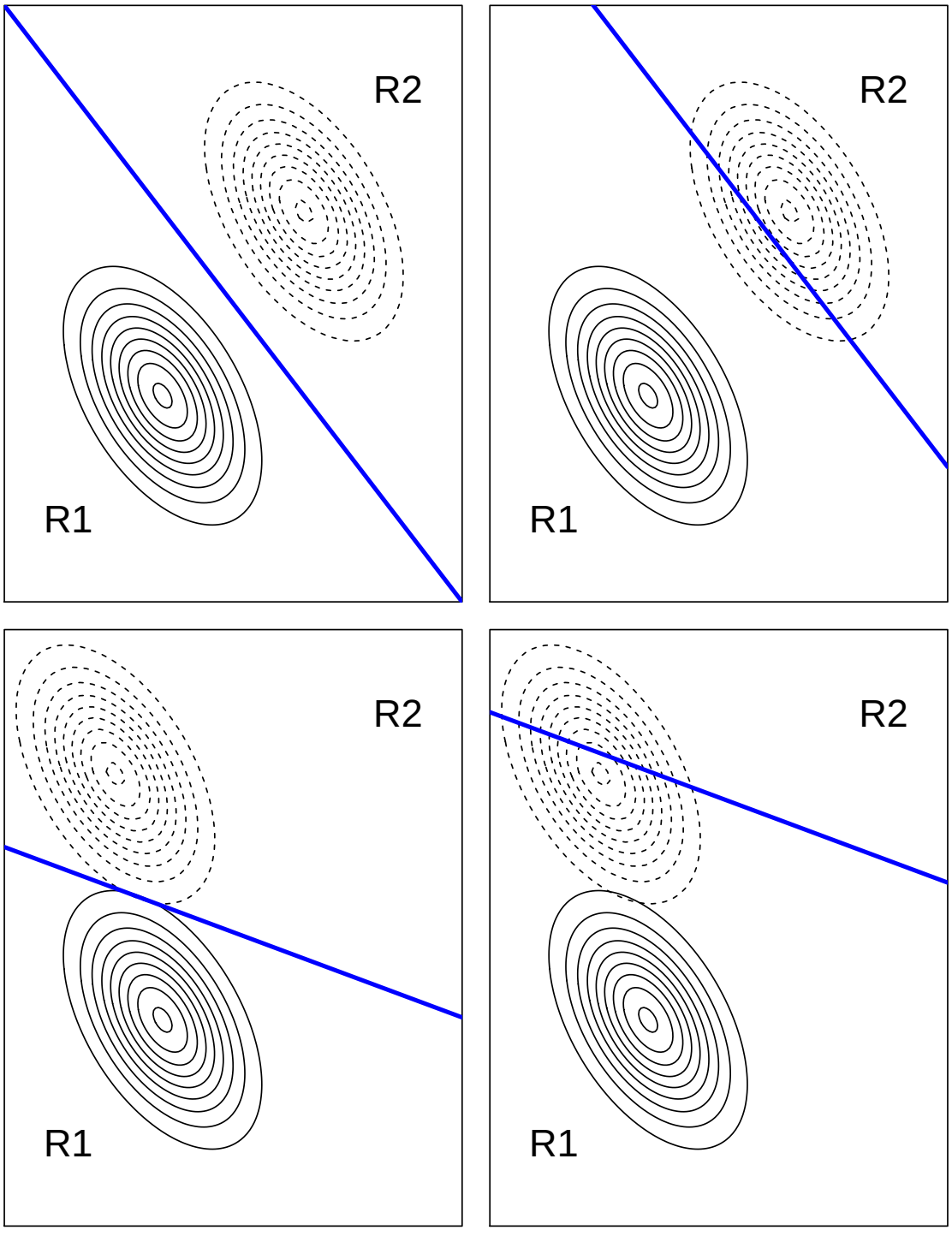



Figure 47: Optimal rule for two Normal populations. Top: $\boldsymbol{\mu}_1 = (0,0)^\mathsf{T}$, $\boldsymbol{\mu}_2 = (3,3)^\mathsf{T}$; Bottom: same $\boldsymbol{\mu}_1$, but $\boldsymbol{\mu}_2 = (-1,4)$, and common $\sigma_{11} = \sigma_{22} = 1$, $\sigma_{12} = -0.5$ across both groups. Left: $\pi_1 = \pi_2$,$c_2 = c_1$; Right: $\pi_1 = 0.9$,$c_2 = c_1$. Notice that the classification boundary is not necessarily aligned with an eigenvector of the covariance, nor is it orthogonal to $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$.

Hence the optimal rule is to assign to group 1 when

$$-\frac{1}{2}\left(-\frac{1}{3}x_1^2 - \frac{1}{3}x_2^2 + \frac{4}{3}x_1x_2\right) + \frac{2}{3}x_1 - \frac{10}{3}x_2 + 4.52$$
$$= \frac{x_1^2}{6} + \frac{x_2^2}{6} - \frac{2}{3}x_1x_2 + \frac{2}{3}x_1 - \frac{10}{3}x_2 + 4.52 > 0.$$

Figure 48 shows the contours of the densities under each group, and the classification regions under the optimal rule.

```
m1 <- c(0,0); m2 <- c(1,3)
S1 <- matrix(c(1,0,0,1),nrow=2)
S2 <- matrix(c(1,.5,.5,1),nrow=2)
B <- solve(S1) - solve(S2)
b <- solve(S1) %*% matrix(m1,ncol=1) - solve(S2) %*% matrix(m2,ncol=1)
k <- .5*(log(det(S2)/det(S1)) + matrix(m2,nrow=1) %*% solve(S2) %*%
         matrix(m2,ncol=1) - matrix(m1,nrow=1) %*%
         solve(S1) %*% matrix(m1,ncol=1))

x1seq <- seq(-10,15,length=10^4)
cf <- cbind(-.5*B[2,2], b[2] - .5*B[1,2]*x1seq, k -.5*B[1,1]*x1seq^2 +
            b[1]*x1seq)
root <- t(apply(cf, 1, function(z) polyroot(rev(z))))
sel <- rowSums(abs(Im(root))< 10^(-5))==2
#
library(mvtnorm)
xseq <- seq(-3,3,length=100)
xgrid <- expand.grid(xseq,xseq)
fx <- dmvnorm(xgrid, mean=m1, sigma=S1)
contour(x=xseq,y=xseq,z=matrix(fx,nrow=length(xseq)), xlim=c(-3,6),
        ylim=c(-3,6),drawlabels=FALSE, asp=1)
xseq <- seq(-6,6,length=100)
xgrid <- expand.grid(xseq,xseq)
fx <- dmvnorm(xgrid, mean=m2, sigma=S2)
contour(x=xseq,y=xseq,z=matrix(fx,nrow=length(xseq)), xlim=c(-3,6),
        ylim=c(-3,6),drawlabels=FALSE,lwd=1,lty=2, add=TRUE)
#
lines(x1seq[sel], root[sel,1], col='blue', lwd=3)
lines(x1seq[sel], root[sel,2], col='blue', lwd=3)
text(-3,0,'R1',cex=1.5); text(-2,5,'R2',cex=1.5)
```

## 6.3 Data-based classifiers and out-of-sample performance

Our results so far assumed that the class-specific densities $f(\mathbf{x} \mid Y = 1)$ and $f(\mathbf{x} \mid Y = 2)$ were known. In practice, these densities are not known and must be inferred

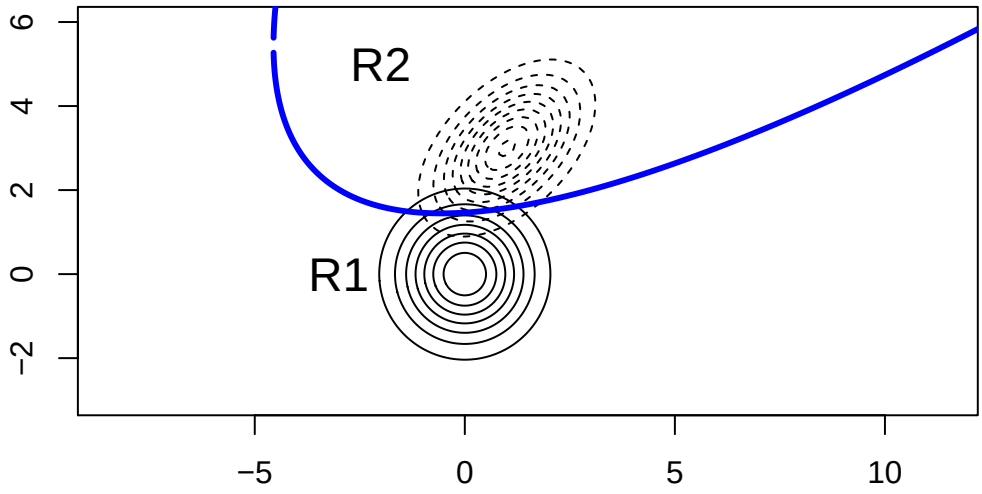



Figure 48: Optimal rule for two Normal populations with $\boldsymbol{\mu}_1 = (0,0)^\mathsf{T}$, $\boldsymbol{\mu}_2 = (1,3)^\mathsf{T}$, common $\sigma_{11} = \sigma_{22} = 1$ and correlation 0 in group 1 and 0.5 in group 2.

from the observed data. This task can be challenging, especially when the predictor $\mathbf{x}$ is high-dimensional. On the one hand, assuming an overly simple parametric form may fail to capture important features, resulting in poor predictions. On the other hand, more flexible or non-parametric estimates may be quite unstable unless the sample size is very large, again resulting in poor predictions.

An implication is that, because we do not know the class-specific densities, the optimal rule given in Proposition 6.1.3 no longer applies. In this situation, one typically compares the performance of several classifiers empirically.

A naive approach to assess correct and incorrect classification rates is to determine the classification regions $R_1, R_2$ from the observed data $(y_1, \mathbf{x}_1^\mathsf{T}), \ldots, (y_n, \mathbf{x}_n^\mathsf{T})$. Then one simply compares the predictions $\hat{y}_1, \ldots, \hat{y}_n$ with the observed $y_1, \ldots, y_n$ to build the so-called *confusion matrix*.

**Definition 6.3.1 (Confusion matrix).**
Let $y_i$ be the observed class for individual $i = 1, \ldots, n$, and let $\hat{y}_i$ be the predicted class. The **confusion matrix** is the following $2 \times 2$ contingency table comparing $y_i$ with $\hat{y}_i$.

| | Observed | |
|---|---|---|
| Predicted | $y = 1$ | $y = 2$ |
| $\hat{y} = 1$ | a | b |
| $\hat{y} = 2$ | c | d |

An obvious and very serious limitation of this approach is that it uses the same data for training the classification algorithm (i.e. determining $R_1$,$R_2$) and evaluating its predictive accuracy. The consequence is that this naive approach will tend to over-estimate the classification power. Even worse, the over-estimation will be more pronounced for complex rules depending on many parameters than for simpler rules, so that we cannot even use this approach to compare rules.

Fortunately, there is an easy alternative based on dividing the observed data into a *training data set* and a *test data set*. The training set is used to define the classification rule, which is then applied to the test data set. Because observations in the test data were not used to calibrate the classifier, there is no possibility of over-fitting. Literally dividing the observed data into two sets is not very convenient, as it requires having a large sample size. The usual approach is to use *leave-one-out cross-validation*, which is based on iteratively leaving one observation out.

**Algorithm 6.3.1** (confusion matrix using leave-one-out cross-validation)**.**

For $i = 1, \ldots, n$ do the following steps:

1. Exclude the $i^{th}$ observation $(Y_i, \mathbf{x}_i^\mathsf{T})$ from the data.
2. Use the remaining data to determine the regions $R_1^{(i)}$,$R_2^{(i)}$.
3. If $\mathbf{x}_i \in R_1^{(i)}$ set $\hat{Y}_i = 1$, else $\hat{Y}_i = 2$.

Build a confusion matrix (Definition 6.3.1) comparing $Y_i$ and $\hat{Y}_i$.

There are other versions of cross-validation, such as $k$-fold cross validation, where the sample is divided into $k$ subsamples of (approximately) equal size, $k-1$ of which are used to train/estimate the classifier $\hat{g}$, and it is then evaluated on the omitted subsample. The procedure is then repeated by leaving out each of the $k$ subsamples. The whole process is repeated for a (sufficiently large) number of random splitting of the data into $k$ groups. Notice that leave-one-out cross-validation is the same as $n$-fold cross-validation.

**Example 6.3.2.** We consider Anderson's Iris data with 4 measurements for multiple flowers, and select only flowers from species Versicolor or Virginica. A Principal Components plot suggests that the density of the data can be reasonably approximated with a multivariate Normal, so we apply the optimal classification rule assuming $\Sigma_1 \neq \Sigma_2$. Here since the number of `Versicolor` and `Virginica` flowers is the same, we set $\pi_1 = \pi_2$. Assume we have no preference in terms of misclassification costs, so $c_2 = c_1$.

The table below shows the confusion matrix, where we used the same data both to train and test the classifier.

| | Versicolor | Virginica |
|---|---|---|
| Versicolor | 48 | 1 |
| Virginica | 2 | 49 |

From the table one would naively estimate that the probability of correct classifications is 97/100. As discussed before, this estimate tends to be over-optimistic, and we should rather use cross-validation.

Below we show the R code required to run the classifier. We define a function `normclassif` that estimates $\boldsymbol{\mu}_1$,$\boldsymbol{\mu}_2$, $\Sigma_1$,$\Sigma_2$ from the training data `x`, and then applies the classifier both to `x` and to test data `xnew` that was not used for the estimation part.

```
normclassif <- function(x, groups, xnew, priorprob=c(.5,.5),
costs=c(1,1)) {
    #2 group classifier based on multiv Normal, unequal covar
    # - x: training set
    # - groups: group labels for training set
    # - xnew: test set
    # - priorprob: P(class 1), P(class 2)
    # - costs: misclassification costs c_2, c_1
    library(mvtnorm)
    if (nrow(x)!=length(groups)) stop("nrow(x) != length(groups)")
    g1 <- unique(groups)[1]
    g2 <- unique(groups)[2]
    m1 <- colMeans(x[groups==g1,])
    S1 <- cov(x[groups==g1,])
    m2 <- colMeans(x[groups==g2,])
    S2 <- cov(x[groups==g2,])
    logf1 <- dmvnorm(x,mean=m1,sigma=S1,log=TRUE)
    logf2 <- dmvnorm(x,mean=m2,sigma=S2,log=TRUE)
    logct <- log(priorprob[1]/priorprob[2]) - log(costs[2]/costs[1])
    class1 <- logf1-logf2 + logct > 0
    xpred <- ifelse(class1,as.character(g1),as.character(g2))
    if (!missing(xnew)) {
        logf1 <- dmvnorm(xnew,mean=m1,sigma=S1,log=TRUE)
        logf2 <- dmvnorm(xnew,mean=m2,sigma=S2,log=TRUE)
        class1 <- logf1-logf2 + logct > 0
        xnewpred <- ifelse(class1,as.character(g1),as.character(g2))
        ans <- list(xpred=xpred, xnewpred=xnewpred)
        } else {
        ans <- list(xpred=xpred, xnewpred=NA)
    }
    return(ans)
}

data(iris)
x <- iris[iris$Species!='setosa',]
groups <- factor(x$Species)
x <- x[,1:4]
```

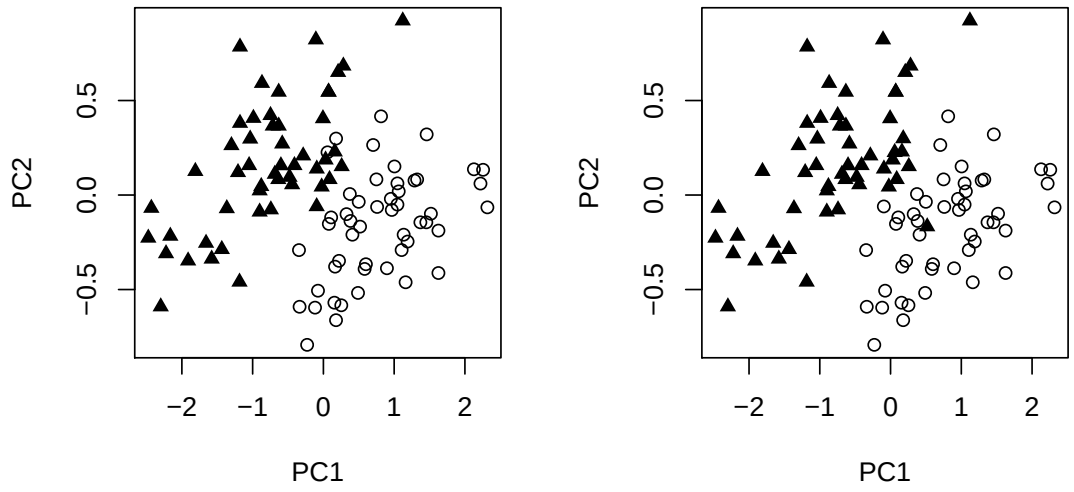



Figure 49: Principal components for Anderson's Iris data, selecting species Versicolor and Virginica. Left: observed data; Right: cross-validated predictions.

```
op <- par(mfrow=c(1,2))
z <- prcomp(x)
pch <- ifelse(as.numeric(groups)==1,1,17)
plot(z$x[,1:2],pch=pch)

pred <- normclassif(x, groups=groups)
table(pred$xpred,groups)

##             groups
##              versicolor virginica
##   versicolor         48         1
##   virginica           2        49
```

With the function `normclassif` in place, we now turn to cross-validation, by iteratively leaving 1 observation out and making a prediction for that observation. We then compare these cross-validated predictions with the true group labels, finding that there were 96/100 correct classifications. In this particular example, cross-validation does not change the estimated correct classification rate by much, but as we shall see in other examples the difference can be quite important. The right panel in Figure 49 shows the cross-validated predictions.

```
pred.cv <- character(nrow(x))
for (i in 1:nrow(x)) {
    fit <- normclassif(x[-i,], groups=groups[-i], xnew=x[i,])
```

```
    pred.cv[i] <- fit$xnewpred
}
table(pred.cv,groups)

##             groups
## pred.cv      versicolor virginica
##   versicolor         47         1
##   virginica           3        49
```

---

## 6.4 Linear Discriminant Analysis

Fisher proposed a classification rule called *linear discriminant analysis* (LDA) which turns out to be equivalent to the Bayes classifier for multivariate Normal data with common covariance matrix across groups and $\pi_1 = \pi_2, c_1 = c_2$. The interesting point is that he arrived to this solution through a completely different argument, and in fact never assumed to have Normal data. LDA arises as the optimal linear classifier in terms of achieving separation between groups.

Let $\mathbf{x}_{1i} \in \mathbb{R}^p$ for $i = 1, \ldots, n_1$ be the observed predictor values in group 1 and $\mathbf{x}_{2i} \in \mathbb{R}^p$ for $i = 1, \ldots, n_2$ be those in group 2. The goal in LDA is to find a linear combination $z_{ji} = \mathbf{a}^\mathsf{T}\mathbf{x}_{ji} \in \mathbb{R}$ such that the ratio of variability between groups relative to the total variability is maximized. That is, we want to find coefficients $\mathbf{a} \in \mathbb{R}^p$ to maximise

$$\frac{(\bar{z}_1 - \bar{z}_2)^2}{s_z^2} = \frac{(\mathbf{a}^\mathsf{T}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2))^2}{\mathbf{a}^\mathsf{T} S_p \mathbf{a}}, \tag{6.4.1}$$

where $\bar{z}_j$ is the sample mean of $(z_{ji})_i$ for $j = 1, 2$, $s_z^2$ is the pooled variance of $z_{ij}$,

$$s_z^2 = \frac{1}{n_1 + n_2 - 2} \sum_{j=1}^{2} \sum_{i=1}^{n_j} (z_{ji} - \bar{z}_j)^2,$$

and $S_p$ the pooled covariance matrix of $\mathbf{x}_{ji}$,

$$S_p = \frac{1}{n_1 + n_2 - 2} \sum_{j=1}^{2} \sum_{i=1}^{n_j} (\mathbf{x}_{ji} - \bar{\mathbf{x}}_j)(\mathbf{x}_{ji} - \bar{\mathbf{x}}_j)^\mathsf{T}.$$

Here we need to assume that $S_p$ is SPD, otherwise the maximum of (6.4.1) would be infinity. This forces us to have $n_1 + n_2 \geq p + 2$. Thus if we have less data points than the dimension of the data, LDA is not applicable. Also, even if there is enough data, it is possible that $S_p$ is not SPD: for instance in the `ZIP code` dataset, taking only digits $0, 1$, the sample size is larger than $p+2$, but the pooled covariance is not SPD:

```r
if(!exists('zip_code'))
{
    zip_code  <- read.table("~/st323/data/zip.train")
    digit <- zip_code[,1] ## the digit label
    zip.mat <- as.matrix(zip_code[,-1])
}

X0 <- zip.mat[digit==0,]
X1 <- zip.mat[digit==1,]
Sp <- ((nrow(X0)-1)*cov(X0) + (nrow(X1)-1)*cov(X1))/
    (nrow(X0) + nrow(X1) - 2)
imin <- Sp %>% diag %>% which.min ##
print(paste0(nrow(X0)+nrow(X1), ' data points in ', ncol(X0),
             ' dimensions'))

## [1] "2199 data points in 256 dimensions"

print(paste0("The minimal entry of the diagonal of the pooled
             covariance is ", Sp[imin, imin])) ##

## [1] "The minimal entry of the diagonal of the pooled
covariance is 0"
```

Figure 50 illustrates the idea behind LDA: we are looking for projections of the data that best separates the two groups.

As we shall see in Proposition 6.4.1 there is a simple expression for $\mathbf{a}$. But let us assume for a moment that $\mathbf{a}$ is known, then the rule is assign a new observation $\mathbf{x}^*$ to group 1 whenever its projection is closer to $\bar{z}_1$ than to $\bar{z}_2$, that is

$$\begin{aligned} & (\mathbf{a}^\mathsf{T}\mathbf{x}^* - \bar{z}_1)^2 < (\mathbf{a}^\mathsf{T}\mathbf{x}^* - \bar{z}_2)^2 \\ \Leftrightarrow \quad & \bar{z}_1^2 - 2\bar{z}_1\mathbf{a}^\mathsf{T}\mathbf{x}^* < \bar{z}_2^2 - 2\bar{z}_2\mathbf{a}^\mathsf{T}\mathbf{x}^* \\ \Leftrightarrow \quad & \bar{z}_1^2 - \bar{z}_2^2 < 2(\bar{z}_1 - \bar{z}_2)\mathbf{a}^\mathsf{T}\mathbf{x}^*. \end{aligned}$$

We would like to simplify the last inequality by dividing by $(\bar{z}_1 - \bar{z}_2)$, but we don't know its sign.... It turns out that this value is positive for $\mathbf{a} \in \mathbb{R}^p$ that maximizes (6.4.1). We have the following result.

**Proposition 6.4.1 (Fisher's Linear Discriminant Analysis).** Assume $\overline{\mathbf{x}}_1 \neq \overline{\mathbf{x}}_2$. The maximizer of (6.4.1) is

$$\mathbf{a} \triangleq S_p^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2),$$

where $S_p$ is the pooled covariance matrix (assumed to be non-singular, i.e. $n_1 + n_2 - 2 \geq p$).

The rule is to allocate $\mathbf{x}^*$ to group 1 if

$$\mathbf{a}^\mathsf{T}\mathbf{x}^* > \frac{1}{2}(\bar{z}_1 + \bar{z}_2) = \frac{1}{2}\mathbf{a}^\mathsf{T}(\overline{\mathbf{x}}_1 + \overline{\mathbf{x}}_2)$$

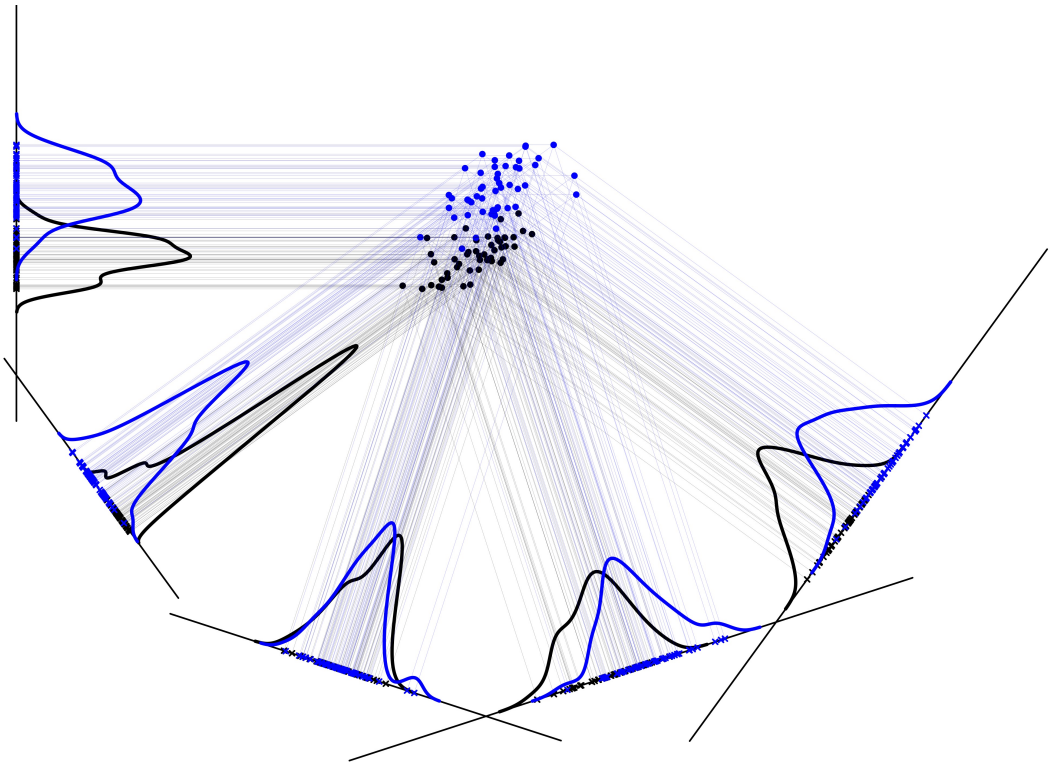


Figure 50: Several 1D projections of part of the `iris` dataset (`versicolor` and `virginica` species), and the corresponding species densities estimates. Notice that some of the projections separate well the two species, whereas some do not separate the species well.

and to allocate to group 2 otherwise.

*Proof.* We first prove the following result: for any symmetric positive definite (SPD) $p \times p$ matrix $S$ and non-zero $\mathbf{u} \in \mathbb{R}^p$, Then for any non-zero $\mathbf{a} \in \mathbb{R}^p$

$$\max_{\mathbf{a} \neq 0} \frac{(\mathbf{a}^\mathsf{T}\mathbf{u})^2}{\mathbf{a}^\mathsf{T} S \mathbf{a}} = \mathbf{u}^\mathsf{T} S^{-1} \mathbf{u},$$

and the maximum is attained for $\mathbf{a} = S^{-1}\mathbf{u}$. Indeed,

$$(\mathbf{a}^\mathsf{T}\mathbf{u})^2 = \left((S^{1/2}\mathbf{a})^\mathsf{T}(S^{-1/2}\mathbf{u})\right)^2 \leq (\mathbf{a}^\mathsf{T} S \mathbf{a})(\mathbf{u}^\mathsf{T} S^{-1} \mathbf{u}),$$

by the Cauchy–Schwartz inequality, and setting $\mathbf{a} = S^{-1}\mathbf{u}$ we get an equality.

Given this result, setting $\mathbf{u} = \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2$, we see that the maximizer of (6.4.1) is given by $\mathbf{a} = S_p^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)$ We now need to find the classification rule: we already know that we classify a new observation $\mathbf{x}^*$ to group 1 if

$$\bar{z}_1^2 - \bar{z}_2^2 < 2(\bar{z}_1 - \bar{z}_2)\mathbf{a}^\mathsf{T}\mathbf{x}^*.$$

Now notice that $\bar{z}_1 - \bar{z}_2 = a^\mathsf{T}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2) = (\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\mathsf{T} S_p^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2) > 0$ since $S_p$ is SPD. Therefore we can simplify the classification rule to "classify to group 1 if

$$\frac{1}{2}(\bar{z}_1 + \bar{z}_2) < \mathbf{a}^\mathsf{T}\mathbf{x}^*$$

and classify to group 2 otherwise."

□

We remark that the rule is equivalent to the one given in Proposition 6.2.1, replacing the means and covariance by their sample estimates and setting $\pi_1 = \pi_2$, $c_2 = c_1$. A nice consequence of this Proposition is that it helps justify the rule based on the multivariate Normal with equal covariances as an optimal linear decision rule. Intuitively, it helps us understand that all we're really doing is transforming the original $\mathbf{x}$ into a scalar value $y$, and then checking whether it's closer to $\bar{y}_1$ or $\bar{y}_2$.

**Example 6.4.2.** Consider the Iris data and apply Fisher's LDA to discriminate between the species Versicolor and Virginica.

$$\overline{\mathbf{x}}_1 = \begin{pmatrix} 5.94 \\ 2.77 \\ 4.26 \\ 1.33 \end{pmatrix}; \overline{\mathbf{x}}_2 = \begin{pmatrix} 6.59 \\ 2.97 \\ 5.55 \\ 2.03 \end{pmatrix}; S_p = \begin{pmatrix} 0.34 & 0.09 & 0.24 & 0.05 \\ 0.09 & 0.10 & 0.08 & 0.04 \\ 0.24 & 0.08 & 0.26 & 0.06 \\ 0.05 & 0.04 & 0.06 & 0.06 \end{pmatrix}$$

The coefficients for the optimal linear combination are therefore

$$\mathbf{a} = S_p^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2) = \begin{pmatrix} 3.56 \\ 5.58 \\ -6.97 \\ -12.38 \end{pmatrix}.$$

Figure 51 shows the projections of the observed data and the midpoint between $\bar{y}_1$ and $\bar{y}_2$, i.e. the mean of the projections from each group. The plot suggests that the linear rule is able to discriminate the two groups pretty well. This is confirmed when we apply cross-validation, which delivers the following confusion matrix.

| | Versicolor | Virginica |
|---|---|---|
| Versicolor | 48 | 1 |
| Virginica | 2 | 49 |

```
##LDA for Iris data
data(iris)
x <- iris[iris$Species!='setosa',]
groups <- factor(x$Species)
x <- x[,1:4]


fisherlda <- function(x, groups, xnew) {
  g1 <- unique(groups)[1]
  g2 <- unique(groups)[2]
  #
  m1 <- colMeans(x[groups==g1,])
  m2 <- colMeans(x[groups==g2,])
  S1 <- cov(x[groups==g1,])
  S2 <- cov(x[groups==g2,])
  n1 <- sum(groups==g1)
  n2 <- sum(groups==g2)
  Sp <- ((n1-1)*S1 + (n2-1)*S2)/(n1+n2-2)
  #
  a <- solve(Sp) %*% matrix(m1-m2,ncol=1)
  z <- as.matrix(x) %*% a
  midpoint <- as.vector(.5*(m1 %*% a + m2 %*% a))
  pred <- z > midpoint
  pred <- ifelse(pred,as.character(g1),as.character(g2))
  ans <- list(z=z,xpred=pred,midpoint=midpoint, a=a)
  if (!missing(xnew)) {
    xnewpred <- (as.matrix(xnew) %*% a) > midpoint
    xnewpred <- ifelse(xnewpred,g1,g2)
    ans$xnewpred <- xnewpred
  } else {
    ans$xnewpred <- NA
  }
  return(ans)
}

lda <- fisherlda(x,groups)
table(lda$xpred,groups)

##             groups
##              versicolor virginica
##   versicolor         48         1
##   virginica           2        49
```

```
pred.cv <- character(nrow(x))
for (i in 1:nrow(x)) {
  fit <- fisherlda(x[-i,], groups=groups[-i], xnew=x[i,])
  pred.cv[i] <- fit$xnewpred
}

table(pred.cv,groups)

##         groups
## pred.cv versicolor virginica
##       1         48         1
##       2          2        49

op <- par(mfrow=c(3,1))
layout(matrix(c(1,2,2), nrow=3))
pch <- ifelse(as.numeric(groups)==1,4,4)
col <- ifelse(as.numeric(groups)==1,'black','blue')
plot(lda$z,rep(1,nrow(x)),col=col,pch=pch,xlab="a'x",
     ylab='',cex.lab=1.25,ylim=c(.9,2),yaxt='n')
abline(v=lda$midpoint,lty=2,lwd=2)


X = iris[iris$Species != 'setosa',]
species=factor(X$Species)
X = X[, c(2,4)]
X = (as.matrix(X))
X <- jitter(X, amount=.05) # we slightly jitter the observations to get a nicer pl
X = scale(X, scale=FALSE)


par(mai=rep(0.,4))
plot(X, cex=.6, asp=1, xlim=c(-5.5, 3), ylim=c(-4, 2), pch=20,
     axes=FALSE, col=c(1,4)[species])
a <- lda$a[c(2,4)]
projection_plot(X, groups=species, a,  ind=NULL,  lwdproj=.05,
                density_scaling=1, shift_len=5)
lines(c(-1,1)*a[2], -c(-1,1)*a[1], lty=2, lwd=2)
par(op)
```

## 6.5 $K$-Nearest Neighbours Classification

Recall that Bayes' classifier can be sometimes written in terms of posterior class probabilities $\mathbb{P}(Y = y|\mathbf{X} = \mathbf{x})$, see Remark 6.1.4. $K$-nearest neighbours (KNN) is a

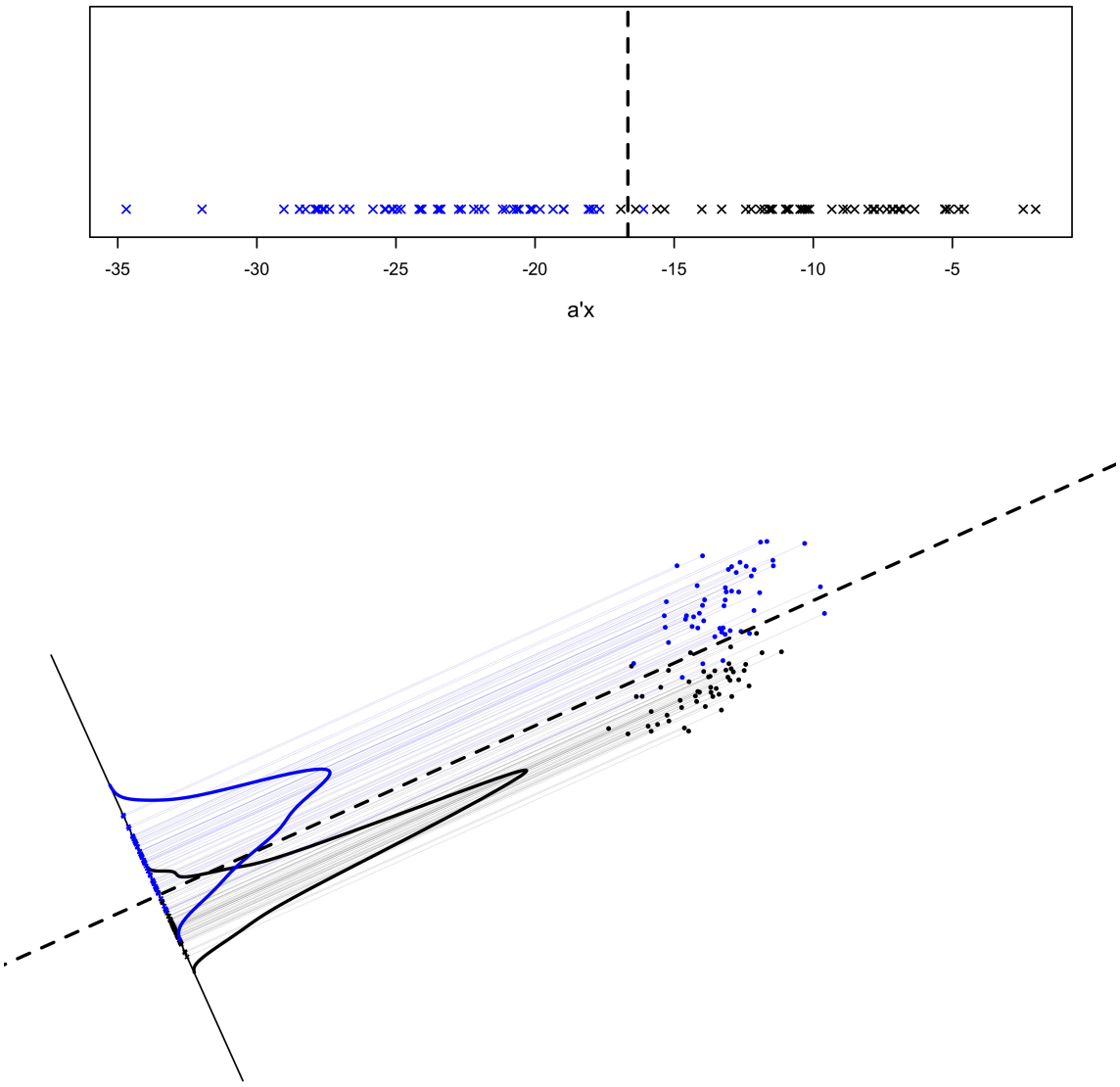



Figure 51: Fisher's Linear Discriminant Rule to classify `versicolor` vs. `virginica`. Top subfigure: the vertical dashed line shows the midpoint between $\bar{y}_1$, $\bar{y}_2$. Bottom subfigure: the corresponding projections of the original data (here we have slightly cheated—to allow for a graphical representation—in that we have taken the projection $\mathbf{x} \mapsto \mathbf{a}^\mathsf{T}\mathbf{x}$ restricted to the variables 2 & 4.

simple but highly intuitive classification technique.

**Algorithm 6.5.1** ($K$-nearest neighbours)**.**
Let $\mathbf{x}$ be the predictor values for an individual we wish to classify, and let $c_2, c_1$ be misclassification costs as in Proposition 6.1.3.

1. Find the $K$ individuals closest to $\mathbf{x}$, where distance can be defined in any convenient manner.
2. Estimate $\mathbb{P}(Y = y \mid \mathbf{x})$ by $\hat{\mathbb{P}}(Y = y \mid \mathbf{x})$, the proportion of individuals in class $y$ amongst the $K$ nearest neighbours of $\mathbf{x}$.
3. Classify as $Y = 1$ if
$$\frac{\hat{\mathbb{P}}(Y = 1 \mid \mathbf{x})}{\hat{\mathbb{P}}(Y = 2 \mid \mathbf{x})} > \frac{c_2}{c_1},$$
and as $Y = 2$ otherwise.

When $c_2 = c_1$, which is done frequently in practice, KNN simply follows a majority voting rule. You walk outside of your house, ask your neighbours' opinion, and go with the majority.

The two key choices for KNN are how to measure distance and how to set $K$ (i.e. how many neighbours are needed). To set $K$, one can try different values, assess their predictive power via cross-validation, and choose the best-performing $K$. As a limitation, the approach uses a fixed $K$ for all individuals, but the optimal $K$ in general can be different depending on $\mathbf{x}$. In regions where all individuals belong to one class a large $K$ may be best, but in regions where individuals quickly transition from class 1 to class 2 a smaller $K$ may be preferable.

Figure 52 shows that classification regions for various values of $K$. Notice that the classification region is more "rough" for smaller values of $K$, and becomes "smoother" for larger values of $K$.

**Example 6.5.1.** We use KNN to classify the Iris data. The R function `knn` in package `class` lets us specify a training set where to train the algorithm and an independent test set. We use a leave-one-out cross-validation setting $K = 1$ neighbours, which achieves 94/100 correct classifications.

```
#KNN for Iris data
library(class)
data(iris)
iris <- iris[iris$Species!='setosa',]
x <- as.matrix(iris[,1:4])
x <- scale(x)
names(x) <- names(iris)[1:4]
y <- as.character(iris$Species)

ypred <- rep(NA,length(y))
```

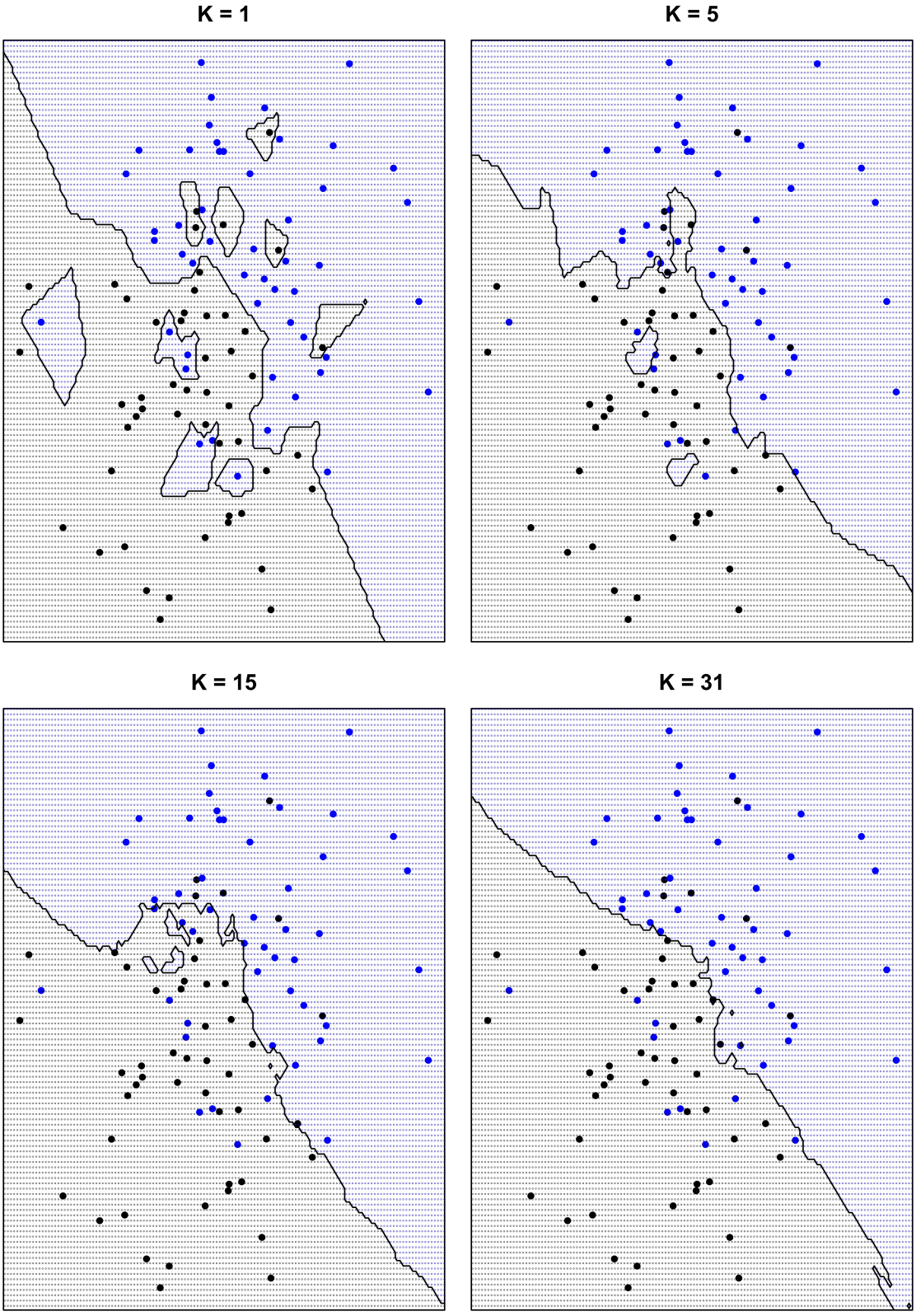



Figure 52: Classification regions for various values of $K$ using $K$-nearest neighbour classification.

```
for (i in 1:length(y)) ypred[i] <- knn(train=x[-i,], test=x[i,],
                                       cl=y[-i], k=1)
table(y,ypred)

##             ypred
## y             1  2
##   versicolor 48  2
##   virginica   4 46

op <- par(mfrow=c(1,2))
col <- ifelse(ypred==1,'black','blue')
pch <- ifelse(ypred==1,16,17)
plot(prcomp(iris[,1:4])$x[,1:2],col=col,pch=pch)


ypred <- rep(NA,length(y))
for (i in 1:length(y)) ypred[i] <- knn(train=x[-i,], test=x[i,],
                                       cl=y[-i], k=18)
table(y,ypred)

##             ypred
## y             1  2
##   versicolor 48  2
##   virginica   2 48

col <- ifelse(ypred==1,'black','blue')
pch <- ifelse(ypred==1,16,17)
plot(prcomp(iris[,1:4])$x[,1:2],col=col,pch=pch)

par(op)
```

We now repeat the analysis with a larger $K = 18$, finding 95/100 correct classifications (cross-validated). The PC plots of the classifications are given in Figure 53.

We assess the performance of the algorithm for $K$ ranging from 1 to 50. For convenience, we define a function `knn.cv` that returns the cross-validated correct classification rate. Figure 54 shows the results.

```
knn.cv <- function(x,y,kmax=20) {
    #KNN correct classification rate for several K (cross-valid)
    cc <- rep(NA,kmax)
    for (k in 1:kmax) {
        ypred <- rep(NA,length(y))
        for (i in 1:length(y)) ypred[i] <- knn(train=x[-i,], test=x[i,],
                                               cl=y[-i], k=k)
        tab <- table(y,ypred)
        cc[k] <- sum(diag(tab))/length(y)
```

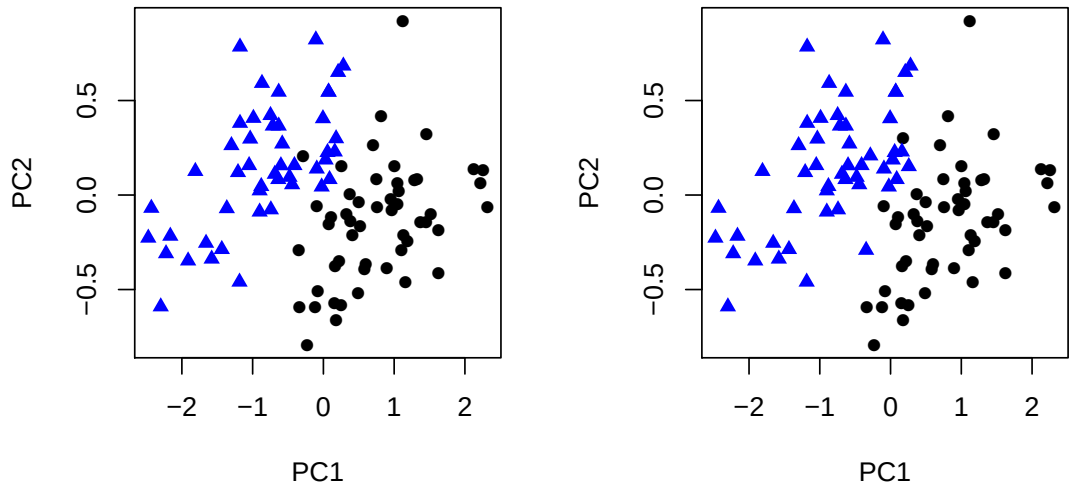



Figure 53: Principal components for the Iris data showing KNN predictions for $K = 1$ (left) and $K = 18$ (right).

```
    }
    return(cc)
}

set.seed(1) # for reproducibility
cc <- knn.cv(x,y,kmax=50)

plot(1:length(cc),cc,type='b', xlab='K', pch=20,
     ylab='Correct classification rate (CV)',
     cex.lab=1.25,cex.axis=1.25)
```

The correct classification rate stays quite stable for up to $K = 35$ neighbours, but then starts to decrease. The optimal values are found at $K = 16, 18$.

---

### 6.5.1 Comparison of 1NN with the Bayes classifier

When studying the performance of classification rules, we often aim to compare them to the Bayes classifier. Since the Bayes classifier has minimal error probability, we use this risk as a benchmark to compare the performance of other, data-driven classifiers against. First, let's get an explicit expression for the risk of the Bayes classifier when $c_1 = c_2$.

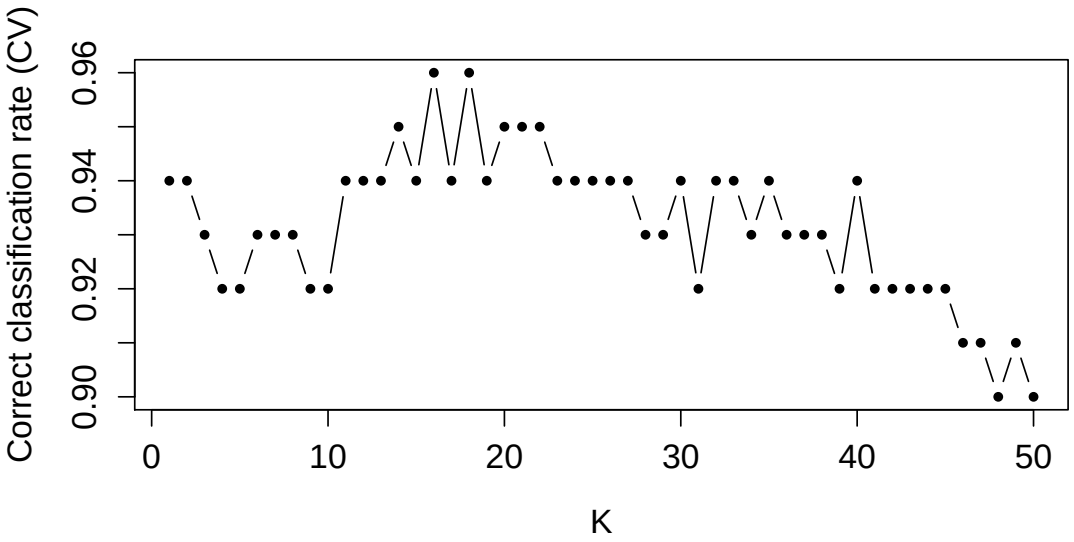



Figure 54: Iris data. Correct classification rate assessed via cross-validation for KNN and $K = 1, \ldots, 50$.

**Proposition 6.5.2.** Suppose $c_1 = c_2$, and let

$$\eta(\mathbf{x}) = \mathbb{P}(Y = 1|\mathbf{X} = \mathbf{x}) = \frac{f(\mathbf{x}|Y = 1)\pi_1}{f(\mathbf{x}|Y = 1)\pi_1 + f(\mathbf{x}|Y = 2)\pi_2},$$

so that the Bayes classifier $g_* : \mathbb{R}^d \to \{1, 2\}$ is given by $g_*(\mathbf{x}) = 1$ if $\eta(\mathbf{x}) > 1/2$ and $g_*(\mathbf{x}) = 2$ if $\eta(\mathbf{x}) \leq 1/2$. Then

$$\mathbb{P}(g_*(\mathbf{X}) \neq Y) = \mathbb{E}[\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}].$$

*Proof.* If $\mathbf{x}$ is such that $\eta(\mathbf{x}) > 1/2$ then we have that

$$\mathbb{P}(g_*(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x}) = \mathbb{P}(Y = 2|\mathbf{X} = \mathbf{x}) = 1 - \eta(\mathbf{x}).$$

On the other hand, if $\mathbf{x}$ is such that $\eta(\mathbf{x}) \leq 1/2$ then we have that

$$\mathbb{P}(g_*(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x}) = \mathbb{P}(Y = 1|\mathbf{X} = \mathbf{x}) = \eta(\mathbf{x}).$$

So, in either case,

$$\mathbb{P}(g_*(\mathbf{X}) \neq Y|\mathbf{X} = \mathbf{x}) = \min\{\eta(\mathbf{x}), 1 - \eta(\mathbf{x})\},$$

and the result follows by taking expectation over $\mathbf{x}$. □

Under mild regularity conditions, we can derive the asymptotic risk of the 1NN classifier and show that is has a similar form to that of the Bayes classifier. This will allow us to compare the performance of the optimal Bayes classifer, which depends

on knowing the distribution of $(\mathbf{X}, Y)$, to the performance of the 1NN classifier, which only requires data. Recall that when $c_1 = c_2$ the 1NN classification rule predicts that the label of $\mathbf{X}$ is the same as the label of the closest data point to it. This is a relatively crude data-driven classifier, but still has some good properties.

**Proposition 6.5.3 (Non-examinable).** Suppose $c_1 = c_2$. Given data $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, let $g_n : \mathbb{R}^p \to \{1, 2\}$ be the 1NN classification rule, so that $g_n(\mathbf{x}) = Y_{\text{NN}}$, where $Y_{\text{NN}}$ is the label of the nearest $\mathbf{X}_i$ to $\mathbf{x}$. Then we have

$$\mathbb{P}(g_n(\mathbf{X}) \neq Y) \to \mathbb{E}[2\eta(\mathbf{X})\{1 - \eta(\mathbf{X})\}]$$

as $n \to \infty$, under mild regularity conditions.

*Proof Sketch, non-examinable.* We have

$$\begin{aligned}
\mathbb{P}(g_n(\mathbf{X}) \neq Y) &= \mathbb{P}(Y_{\text{NN}} \neq Y) = \mathbb{E}\big[\mathbb{P}(Y_{\text{NN}} \neq Y | \mathbf{X})\big] \\
&= \mathbb{E}\big[\eta(\mathbf{X})\mathbb{P}(Y_{\text{NN}} = 2) + \{1 - \eta(\mathbf{X})\}\mathbb{P}(Y_{\text{NN}} = 1)\big] \\
&= \mathbb{E}\big[\eta(\mathbf{X})\{1 - \eta(\mathbf{X}_{\text{NN}})\} + \{1 - \eta(\mathbf{X})\}\eta(\mathbf{X}_{\text{NN}})\big],
\end{aligned}$$

where we write $\mathbf{X}_{\text{NN}}$ for the nearest value of $\mathbf{X}_i$ to $\mathbf{X}$. As $n \to \infty$, we have $\mathbf{X}_{\text{NN}} \xrightarrow{p} \mathbf{X}$, as data points become more tightly packed. So, we also have $\eta(\mathbf{X}_{\text{NN}}) \xrightarrow{p} \eta(\mathbf{X})$ and hence

$$\mathbb{P}(g_n(\mathbf{X}) \neq Y) \to \mathbb{E}[2\eta(\mathbf{X})\{1 - \eta(\mathbf{X})\}].$$

□

We now compare these two expected costs. Write $C_*$ for the expected cost of the Bayes classifier, and write $C$ for the asymptotic expected cost of the 1NN classifier. We will use the fact that, for any $a, b \in \mathbb{R}$, we have $ab = \min(a, b) \times \max(a, b)$. Now

$$\begin{aligned}
C/2 &= \mathbb{E}[\eta(\mathbf{X})\{1 - \eta(\mathbf{X})\}] \\
&= \mathbb{E}\big[\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\} \max\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}\big] \\
&= \mathbb{E}\big[\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}\big(1 - \min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}\big)\big] \\
&= \mathbb{E}\big[\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}\big] - \mathbb{E}\big[\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}^2\big] \\
&= C_* - \left\{\text{Var}\big(\mathbb{E}\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}\big) + \big(\mathbb{E}\big[\min\{\eta(\mathbf{X}), 1 - \eta(\mathbf{X})\}\big]\big)^2\right\} \\
&\leq C_* - C_*^2.
\end{aligned}$$

We therefore have that $C \leq 2C_*(1 - C_*)$. In particular, when we have a lot of data, the expected cost of the 1NN classifier is no worse than twice the optimal expected cost, whatever the distribution of the data is.

**Advantages of KNN**

- Only requires distances, making it a very general algorithm,
- Can capture non-linear and non-monotonic patterns,
- Can detect interactions (complex combinations of variables associated with one of the classes).

**Disadvantages of KNN**

- Requires choosing a distance metric,
- Sub-optimal to detect monotonic patterns,
- Requires a good choice for $K$.

## 6.6 Classification and Regression Trees (CART)

Classification trees provide a very different approach from the methods we have seen so far. Following ideas similar to divisive clustering (which we will see later), initially all individuals are in a single group, which forms the root of the tree. Then that group is split into two nodes, typically by setting a threshold on one of the predictors. Both the predictor and the threshold are usually chosen so that they separate individuals from both classes as much as possible. The obtained nodes are split iteratively, each time using any adequate variable and threshold, until nodes are pure in the sense that they mostly contain observations from a single class.

**Example 6.6.1.** Figure 55, shows a tree classifying prostate cancer patients into progressed / not progressed using multiple predictors (tumor grade, % cells in $G_2$ phase, ploidysm, age etc.). Patients with tumor grade $< 2.5$ are assigned to the 'not progressed' group. For the remaining patients, the tree checks if the value of $G_2$ is below 13.2, and then keeps checking variables until a node is reached.

Figure 56 (left panel) shows the age and % of $G_2$ cells for grade 3-4 patients, as well as the progression / no progression status. The right panel shows the CART predictions, where we added lines defining the classification regions. We see that the space is partitioned in a discontinuous fashion. We note that patients with $G_2 < 13.2$ are also classified according to the variable `ploidy`, which explains the presence of mixed predictions in that region.

```
#CART for Stage C data

library(rpart)
progstat <- factor(stagec$pgstat, levels = 0:1, labels = c("No", "Prog"))
cfit <- rpart(progstat ~ age + eet + g2 + grade + gleason + ploidy,
              data = stagec, method ='class')
y <- progstat
ypred <- predict(cfit)[,2]>0.5

op <- par(xpd=NA)
plot(cfit)
text(cfit)
par(op)
```

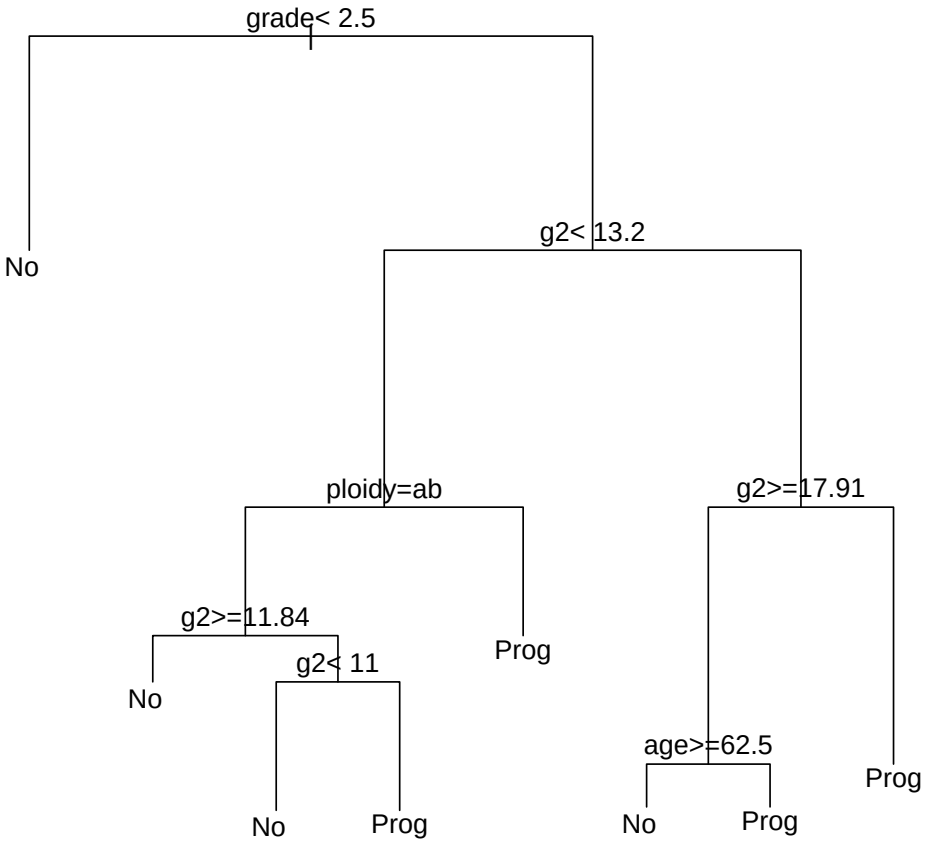



Figure 55: Classification tree for Stage C data

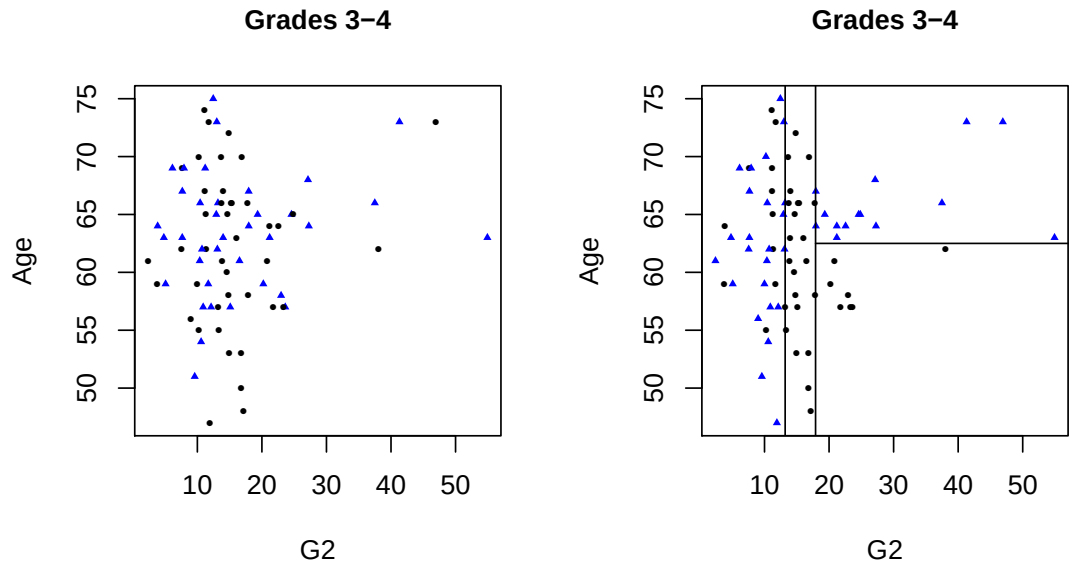



Figure 56: Stage C data for Grade 3-4 patients and progression/no progression status. Left: true status; Right: CART predictions.

```
op <- par(mfrow=c(1,2))
sel <- (stagec$grade>=2.5)
col <- ifelse(y=='Prog','black','blue')
pch <- ifelse(y=='Prog',16,17)
plot(stagec$g2[sel],stagec$age[sel],xlab='G2',ylab='Age',cex.lab=1,
     col=col[sel],pch=pch[sel],main='Grades 3-4',cex.main=1, cex=.5)
#
sel <- (stagec$grade>=2.5)
col <- ifelse(ypred==1,'black','blue')
pch <- ifelse(ypred==1,16,17)
plot(stagec$g2[sel],stagec$age[sel],xlab='G2',ylab='Age',cex.lab=1,
     col=col[sel],pch=pch[sel],main='Grades 3-4',cex.main=1, cex=.5)
abline(v=c(13.2,17.91),lwd=1)
segments(x0=17.91,x1=100,y0=62.5,lwd=1)
#plot(stagec$g2[!sel],stagec$age[!sel],xlab='G2',ylab='Age',
#cex.lab=1.25,col=col[!sel],pch=pch[!sel],main='Grades 1-2',cex.main=1.5)
par(op)
```

---

The main choices that one should consider carefully when building a classification tree is how to select variables and set thresholds at each step, and how to define a stop criterion. There are many strategies for each of these choices, which are too extensive to cover here.

**Example 6.6.2.** We applied CART to Anderson's Iris data, obtaining the tree depicted in Figure 57. The tree simply predicts species `versicolor` when `Petal.Width` $< 1.75$, and `virginica` otherwise.

Figure 58 shows the true and predicted species. In this example CART seems to perform a bit worse than other classifiers. It is interesting to note that in these data there seems to be a smooth transition between groups as we move through the predictor space. Smooth transitions can be hard to capture with hard-thresholding rules.

```
#CART for Iris data
library(rpart)
data(iris)
iris <- iris[iris$Species!='setosa',]
x <- as.matrix(iris[,1:4])
names(x) <- names(iris)[1:4]
y <- factor(iris$Species, labels=0:1)

fit <- rpart(y ~ x,method='class')

op <- par(xpd=NA)
plot(fit)
text(fit)
par(op)
```

```
ypred <- predict(fit)[,2]>0.5
table(y,ypred)

##    ypred
## y   FALSE TRUE
##   0    49    1
##   1     5   45

op <- par(mfrow=c(1,2))
col <- ifelse(y==1,'black','blue')
pch <- ifelse(y==1,16,17)
plot(x[,c('Petal.Width','Petal.Length')],pch=pch,col=col, cex=.6)
#
#
col <- ifelse(ypred==1,'black','blue')
pch <- ifelse(ypred==1,16,17)
plot(x[,c('Petal.Width','Petal.Length')],pch=pch,col=col, cex=.6)
par(op)
```

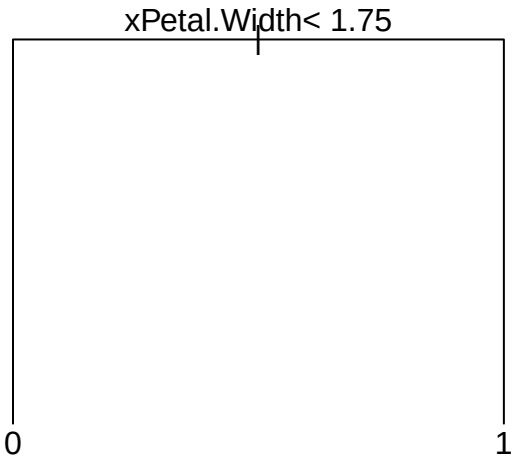



Figure 57: Classification tree for Iris data

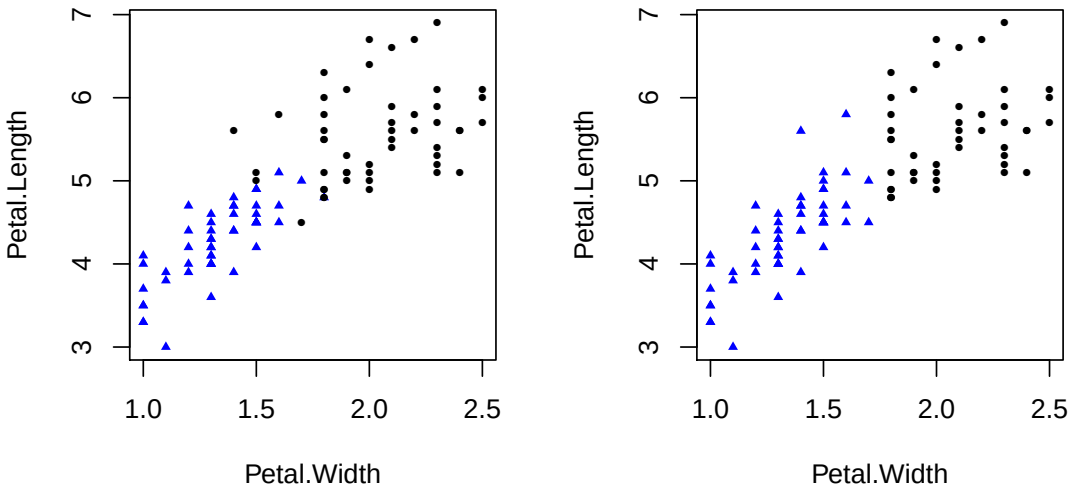



Figure 58: Iris data. Left: true species; Right: CART predictions.

**Advantages of CART**

- Easy to interpret,
- Allows continuous and discrete predictors,
- Captures complex non-linear and non-monotonic patterns.

**Limitations**

- Requires many parameters, and can be sensitive to their choices,
- Sub-optimal to detect monotonic patterns (loss of information due to categorizing),
- Hard to detect interactions (e.g. divisions based on combinations of variables).

## 6.7 Logistic Regression Classification

Logistic regression is a model-based approach to classification that extends the ideas of linear regression. Let $\mathbf{X}_i \in \mathbb{R}^p$ be the vector of explanatory variables (predictors) for individual $i = 1, \ldots, n$, and $Y_i \in \{0, 1\}$ be the binary variable indicating the group of individual $i$ (the $Y_i$s are assumed to be conditionally independent given the $\mathbf{X}_i$s). Notice here that the groups are now labeled $0, 1$ and not $1, 2$, and the cost of misclassifications are denoted by $c_0, c_1 > 0$. This is to allow $Y_i|X_i$ to be modeled as a Bernoulli random variable. Further, let $p_i \triangleq \mathbb{P}(Y_i = 1 \mid \mathbf{X}_i)$ be the probability of class 1, which is allowed to depend on $\mathbf{X}_i$. As in linear regression, the idea is to express the effect of $\mathbf{X}_i$ on the conditional expected value of $Y_i$, $p_i \triangleq \mathbb{E}[Y_i|\mathbf{X}_i]$ using a linear combination of the elements in $\mathbf{X}_i$. However, since $p_i \in (0, 1)$ we cannot model $p_i$ as a linear function of $\mathbf{X}_i$, as the latter can return values outside of $(0, 1)$. Instead, we define the log-odds

$$\eta_i = \log\left(\frac{p_i}{1 - p_i}\right). \tag{6.7.1}$$

The trick is that $\eta_i$ can take any value within the real line, and hence it is reasonable to model $\eta_i$ as a linear function of $\mathbf{X}_i$. That is,

$$\eta_i = \beta_0 + \mathbf{X}_i^\intercal \boldsymbol{\beta} = \beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j,$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$. This model is a particular instance of a generalized linear model (GLM), with the logit link function (6.7.1).

Within this model the problem is reduced to estimating the parameters $\beta_0$ and $\boldsymbol{\beta}$, which in turn produce an estimate for $\eta_i$ and ultimately for $p_i$. Solving for $p_i$ in (6.7.1) gives

$$p_i = \frac{e^{\beta_0 + \mathbf{X}_i^\intercal \beta}}{1 + e^{\beta_0 + \mathbf{X}_i^\intercal \beta}}. \tag{6.7.2}$$

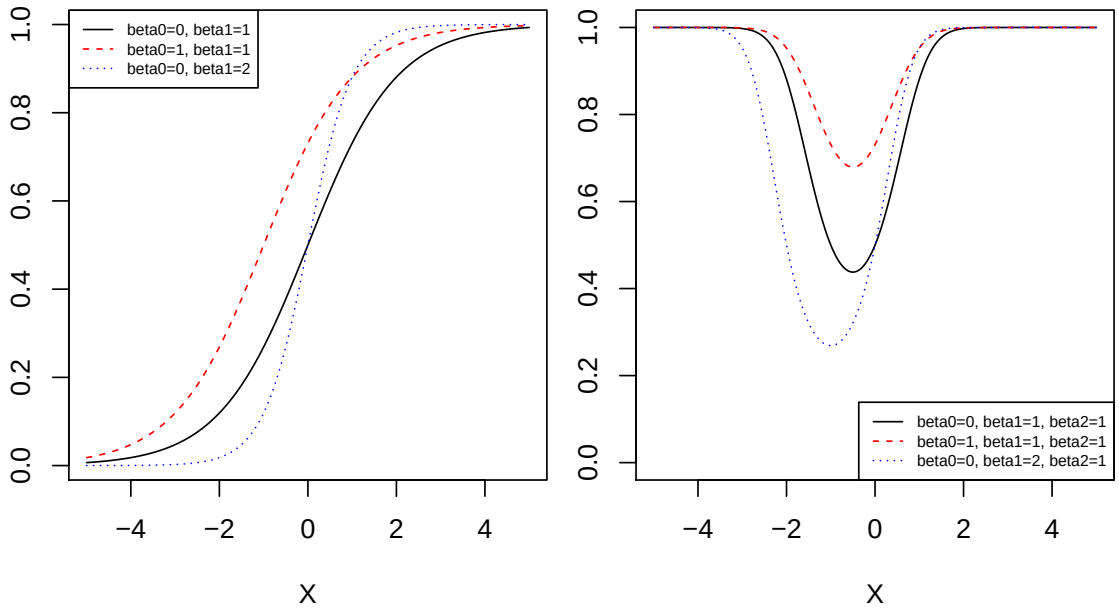



Figure 59: $\mathbb{P}(Y = 1|X)$ as given by logistic regression. Left: $\eta_i = \beta_0 + \beta_1 X_{1i}$; Right: $\eta_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{1i}^2$

Figure 59 (left panel) shows $p_i$ as a function of a single predictor for various values of $\beta_0$ and $\beta_1$. We obtain a sigmoid curve that is monotonic. As with linear regression, the model is quite flexible in that we can include non-linear terms. For instance, we can define $X_2 = X_1^2$ and include it as a new predictor in the equation. Introducing such a quadratic term with $\beta_2 = 1$ in Figure 59 we obtain the curves shown in the right panel. The relationship between $p_i$ and $\mathbf{X}_i$ is no longer monotonic.

The main task is therefore to estimate $\beta_0$ and $\boldsymbol{\beta}$ given a set of observed data. Fortunately, there are efficient algorithms to find maximum likelihood estimates, and asymptotic results (as $n \to \infty$) indicating that these estimates are efficient from a statistical point of view. We shall not discuss the algorithm here, but we will note that it simply iterates weighted least squares steps, and that it is implemented in the R function `glm`.

**Algorithm 6.7.1** (Classification with logistic regression)**.**

1. Obtain estimates $\hat{\beta}_0$,$\hat{\boldsymbol{\beta}}$,
2. For an individual with observed predictors $\mathbf{x}$ compute

$$\hat{\mathbb{P}}(Y = 1 \mid \mathbf{x}) = \left(1 + e^{-\hat{\beta}_0 - \mathbf{x}^\intercal \hat{\boldsymbol{\beta}}}\right)^{-1},$$

3. Let $c_2, c_1$ be the misclassification costs in Proposition 6.1.3. Classify

as $Y = 1$ if

$$\frac{\hat{\mathbb{P}}(Y = 1 \mid \mathbf{x})}{\hat{\mathbb{P}}(Y = 0 \mid \mathbf{x})} > \frac{c_0}{c_1} \iff \mathbf{x}^\intercal \hat{\boldsymbol{\beta}} + \hat{\beta}_0 > \log\left(\frac{c_0}{c_1}\right),$$

and as $Y = 0$ otherwise.

The rule in Algorithm 6.7.1 is very similar to the optimal rule for multivariate Normal data with equal covariances (Proposition 6.2.1), and by extension to Fisher's LDA. The decision is based on a linear combination of the predictors (namely, $\mathbf{x}^\intercal \hat{\boldsymbol{\beta}}$), with an intercept term and a term incorporating the misclassification costs.

In spite of these similarities, the rules are not identical. First, the linear combination coefficients are now given by $\hat{\boldsymbol{\beta}}$, which is not a linear function of the training data (as was the case for the multivariate Normal). Second, logistic regression allows us to include quadratic or any other non-linear transformation of the predictor, and we can even work with binary or categorical predictors (as we would do in linear regression).

**Example 6.7.1.** We look again at the Iris data, where we start by fitting a logistic regression model using function `glm`. This function also fits Normal linear regression models and a wider class of models called Generalized Linear Models. We indicate that we wish to fit a logistic regression model with the argument `family`.

```
data(iris)
iris <- iris[iris$Species!='setosa',]
x <- as.matrix(iris[,1:4])
y <- ifelse(iris$Species=='virginica',1,0)

glm1 <- glm(y ~ x[,1]+x[,2]+x[,3]+x[,4], family=binomial(link='logit'))
summary(glm1)

##
## Call:
## glm(formula = y ~ x[, 1] + x[, 2] + x[, 3] + x[, 4],
##                       family = binomial(link = "logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.01105  -0.00541  -0.00001   0.00677   1.78065
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -42.638     25.707  -1.659   0.0972 .
## x[, 1]        -2.465      2.394  -1.030   0.3032
## x[, 2]        -6.681      4.480  -1.491   0.1359
## x[, 3]         9.429      4.737   1.991   0.0465 *
## x[, 4]        18.286      9.743   1.877   0.0605 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.629  on 99  degrees of freedom
## Residual deviance:  11.899  on 95  degrees of freedom
## AIC: 21.899
##
## Number of Fisher Scoring iterations: 10
```

The `glm` output reveals that we may not need the four predictors (which makes sense, as we know they are highly correlated with each other). We decide to drop the predictor with largest p-value and re-fit the model.

```
glm2 <- glm(y ~ x[,2]+x[,3]+x[,4], family=binomial(link='logit'))
summary(glm2)

##
## Call:
## glm(formula = y ~ x[, 2] + x[, 3] + x[, 4], family = binomial(link = "logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.75795  -0.00412   0.00000   0.00290   1.92193
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -50.527     23.995  -2.106   0.0352 *
## x[, 2]        -8.376      4.761  -1.759   0.0785 .
## x[, 3]         7.875      3.841   2.050   0.0403 *
## x[, 4]        21.430     10.707   2.001   0.0453 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.629  on 99  degrees of freedom
## Residual deviance:  13.266  on 96  degrees of freedom
## AIC: 21.266
##
## Number of Fisher Scoring iterations: 10
```

We will keep the three remaining predictors, as all of them have coefficients minimally different from 0 (although not all of them are strictly significant at the 0.05 level).

Based on this output, and assuming equal misclassification costs $c_2 = c_1$, we compute the probability of each class and assign each individual to the most likely class.

```
b0 <- coef(glm2)[1]
b1 <- matrix(coef(glm2)[-1],ncol=1)
p <- 1/(1+exp(-b0 -x[,2:4] %*% b1))
ypred <- p>0.5

table(y,ypred)

##    ypred
## y   FALSE TRUE
##   0    48    2
##   1     1   49

op <- par(mfrow=c(1,2))
col <- ifelse(ypred==1,'black','blue')
pch <- ifelse(ypred==1,16,17)
plot(prcomp(x)$x[,1:2],pch=pch,col=col, cex=.7)
#
col <- ifelse(y==1,'black','blue')
pch <- ifelse(y==1,16,17)
plot(prcomp(x)$x[,1:2],pch=pch,col=col, cex=.7)
par(op)
```

The confusion matrix in the observed data is similar to those for the multivariate Normal and Fisher's LDA rules. As discussed before, it would be better to obtain the confusion matrix with cross-validation, but we do not do that here. Figure 60 displays the first two principal components and the predicted classes.

---

**Advantages of logistic regression**

- Can detect variables with no predictive power ($\beta_j = 0$),
- Can combine discrete and continuous predictors, and their non-linear transformations,
- No distributional assumptions on predictors $\mathbf{X}$.

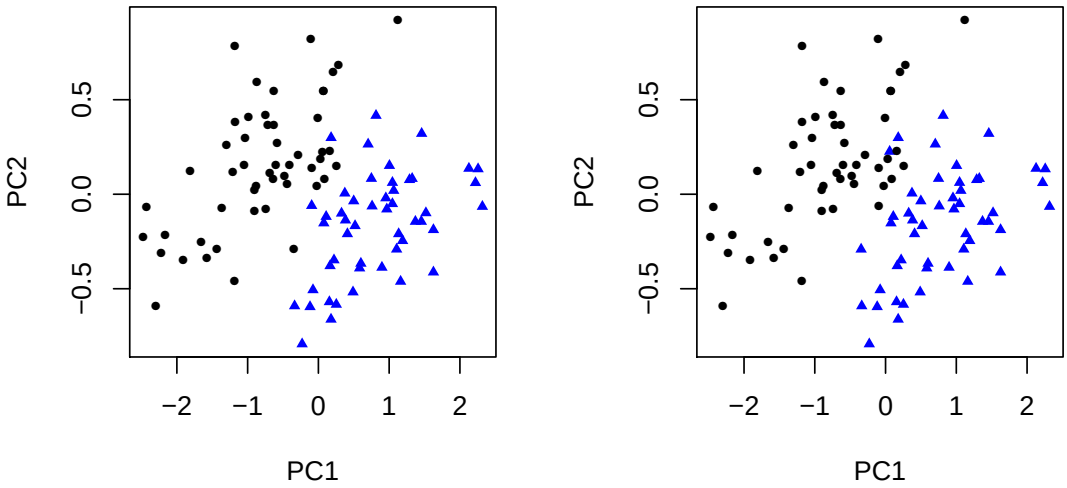



Figure 60: Principal component plot for Iris data (Virginica/Versicolor). Left: logistic regression prediction; Right: true species

# 7 Clustering (Unsupervised Learning)

Clustering algorithms try to detect groups of similar individuals or similar variables, the exact meaning of similarity depending on the application. This task is performed in an unsupervised manner. We are not told which groups exist in the data (if any), rather we want to explore the existence of such groups within the available data.

The output of clustering analysis can help discover underlying structure in the data. For instance, we might find that colon cancer patients are sub-divided into two (previously unknown) groups. Because these groups have different characteristics, they might benefit from different therapies, have different prognosis etc. Another use of clustering is to simplify the interpretation of the data. For instance, a company may record the characteristics of its customers to design effective marketing strategies. Although every customer is different, it would be useful to determine the main customer stereotypes and devise a different strategy for each stereotype. Now consider an example where the goal is to cluster variables. Suppose that a certain supermarket records the amount spent on each item in the basket (so that we have 1 variable per item). As there are many items to be considered, it may be simpler to cluster variables into groups such as fresh food, frozen food, canned food, electronics etc.

From these examples it should be clear that clustering is a technique applicable to virtually any data set. In fact, it has received attention in many fields, including statistics, computer science, machine learning, engineering, bioinformatics etc. For this reason, there is a wide variety of available clustering algorithms, which are impossible to cover in a few lectures. We shall focus on the main philosophical issues and see some popular clustering algorithms that can serve as the basis for more sophisticated strategies.

The fundamental issues in clustering are defining a measure of similarity (or distance), coming up with a strategy to group objects, and assessing the reliability and robustness of the obtained clusters.

## 7.1 Measuring distances

The first step in any clustering analysis is to define a sensible measure of distance or similarity between the items we wish to group (individuals or variables). This choice depends on the specific application, and should be discussed with experts with subject-matter knowledge. A common strategy is to consider several metrics, obtain results for each and evaluate their relative merits. This said, we now discuss some default distance metrics that often prove useful in applications.

We shall distinguish metrics that measure similarity between individuals and those that that measure similarity between variables. Let $X$ be our data matrix, with rows corresponding to individuals $i = 1, \ldots, n$ and columns to variables $j = 1, \ldots, p$. We denote by $\mathbf{x}_{i\cdot}$ the $i^{th}$ row, $\mathbf{x}_{\cdot j}$ the $j^{th}$ column and $x_{ij}$ the $(i, j)$ element in $X$.

When data are real-valued, some common distances between individuals $\mathbf{x}_{i\cdot}$ and $\mathbf{x}_{k\cdot}$ are:

**Euclidean:** $\sqrt{\sum_{j=1}^{p}(x_{ij} - x_{kj})^2}$

**Manhattan:** $\sum_{j=1}^{p} |x_{ij} - x_{kj}|$

**Minkowski:** $\left(\sum_{j=1}^{p} |x_{ij} - x_{kj}|^m\right)^{1/m}$

**Mahalanobis:** $\sqrt{(\mathbf{x}_{i.} - \mathbf{x}_{k.})^\mathsf{T} S^{-1} (\mathbf{x}_{i.} - \mathbf{x}_{k.})}$, where $S = \mathrm{Cov}(\mathbf{x}_{i.}) = \mathrm{Cov}(\mathbf{x}_{k.})$

**Correlation:** $\sqrt{\frac{1}{2}(1 - \rho(\mathbf{x}_{i.}, \mathbf{x}_{k.}))}$, where $\rho(\cdot)$ is some correlation coefficient (Pearson, Spearman, Kendall, etc.).

**Absolute correlation:** $\sqrt{1 - |\rho(\mathbf{x}_{i.}, \mathbf{x}_{k.})|}$

Clearly, the Euclidean and Manhattan distances are particular cases of the Minkowski distance for $m = 2$ and $m = 1$, respectively. The Mahalanobis distance is attractive in that it incorporates correlations between variables. For instance, if two variables had correlation close to 1, they would receive much larger weight (double, roughly) in the Euclidean or Manhattan distances than in the Mahalanobis distance. A difficulty is that it requires knowing $S$, which should be the within-groups covariance, but the groups are of course not known. A way to bypass this difficulty is to perform a first clustering algorithm with Minkowski distances to define groups, then use these groups to estimate $S$, and finally repeat the clustering with Mahalanobis distances.

While Minkowski and Mahalanobis distances can also be used to measure similarity between variables, in practice it is more common to use correlation or association-based metrics.

When variables are categorical, similarities are usually based on contingency tables. Suppose we have the following table comparing rows (or columns) $i$ and $k$

| | | $\mathbf{x}_{i.}$ | |
|---|---|---|---|
| | | 0 | 1 |
| $\mathbf{x}_{k.}$ | 0 | a | b |
| | 1 | c | d |

The following metrics are common choices:

**Proportion of concordance:** $s_{ik} = \frac{a+d}{a+b+c+d}$

**Tanimoto:** $s_{ik} = \frac{d}{b+c+d}$

**Proportion of 1-1 matches:** $s_{ik} = \frac{d}{a+b+c+d}$

**Chi-square test statistic for independence**

The first three measure similarity, while the latter measures dissimilarity.

So far we have seen several metrics to measure either similarity or distance. Obviously, we can always convert a measure of similarity to one of *dissimilarity* (and vice versa). For instance we could define $d_{ik} = 1 - s_{ik}/\max_{j,l} s_{jl}$ or $d_{ik} = 1/s_{ik}$ (assuming that $s_{ik} > 0$). However, there is no guarantee that these transformations define a proper *distance* in the mathematical sense. Clearly both definitions of $d_{ik}$

satisfy $d_{ik} > 0$ and symmetry (if $s_{ik}$ is symmetric), hence the issue is that the triangular inequality may not hold. Gower demonstrated that when the matrix with similarities is non-negative definite and has diagonal entries $s_{ii} = 1$ for all $i = 1, \ldots, n$ the choice $d_{ik} = \sqrt{2(1 - s_{ik})}$ is a proper distance.

In practice, clustering applications often use dissimilarity metrics that do not possess the properties of a distance and obtain reasonable answers. However, thinking about distances adds some elegance, as then the objects being clustered can be thought of as points in some metric space.

## 7.2 Hierarchical clustering

Hierarchical clustering is one of the most popular choices, and includes a family of algorithms. There are two general approaches to hierarchical clustering:

1. Agglomerative (or bottom-up) methods. At the first iteration each object is considered a separate cluster. Then clusters are successively merged until only 1 cluster containing all objects is left.

2. Divisive (or top-down) methods. At the first iteration all objects are included in a single cluster. Then clusters are successively divided until each cluster contains a single object or satisfies some 'purity' criteria (i.e. all observations in the cluster are very similar to each other).

Here we focus on agglomerative hierarchical clustering, which seems to be most popular. The general algorithm is as follows.

---

**Algorithm 7.2.1** (Agglomerative hierarchical clustering)**.**
Let $o_1, \ldots, o_n$ be the objects to be clustered. Denote by

$$C_1^{(0)} \triangleq \{o_1\}, C_2^{(0)} \triangleq \{o_2\}, \ldots, C_n^{(0)} \triangleq \{o_n\},$$

the initial clusters. Set $D^{(0)}$ to be the distance matrix between the initial clusters $\{C_i^{(0)}\}_i$.

There are $n - 1$ steps in the algorithm: For $k = 1, 2, \ldots, n - 1$,

(a) Find the two closest clusters, i.e. find $i \neq j$ such that $(D^{(k-1)})_{ij}$ is minimal, and denote this minimal distance by $d^{(k)}$, i.e.

$$d^{(k)} \triangleq \min_{i \neq j} D_{ij}^{(k-1)},$$

(b) Merge the two closest clusters $C_i^{(k-1)}, C_j^{(k-1)}$ together. Thus redefine the new clusters $C_1^{(k)}, \ldots, C_{n-k}^{(k)}$, and compute the new distance matrix $D^{(k)}$ containing the pairwise distances between these new clusters.

---

Steps (b) in the algorithm requires computing distances between clusters, which requires us to choose a distance between clusters that have more than 1 element.

The method to compute distances between clusters is called the *linkage method*. Four common choices are:

**Single linkage** the distance between two clusters is defined as the distance between the two closest elements in those clusters. Single linkage is unaffected by monotonic changes in the input distances. Because it links nearest neighbours, it can produce elongated clusters where some elements are very far apart from each other.

**Complete linkage** the distance between two clusters is defined as the distance between the two farthest elements in those clusters. Complete linkage is also unaffected by monotonic changes in the input distances. It enforces clusters where no two elements are very different from each other.

**Average linkage** the distance between two clusters is defined as the average of all pairwise distances between those clusters. Average linkage is affected by any non-linear change in the input distances, e.g. working with log-distances produces a different result. It enforces clusters where most elements are nearby, and penalizes clusters where two elements are very far from each other.

**Ward linkage** the distance between two clusters is given by the increase in within-cluster sum-of-squares after merging the clusters. That is, the two clusters to be merged are those minimizing the increase in sum-of-squares. Ward linkage is affected by non-linear changes in the input distances, and prefers spherical clusters.

Figure 61 shows an example with two clusters, both with 3 elements. Single linkage finds the distance between the two closest elements, complete linkage between the two furthest elements, and average linkage averages all pairwise distances.

The result of hierarchical clustering is typically visualized using a dendrogram (from Greek *dendron* 'tree' and *gramma* 'drawing'). The dendrogram is simply a tree whose leaves correspond to individual elements and whose branches indicate which elements/clusters were merged at each step. The $y$-axis indicates the distance between the pair of clusters merged at each stage, which gives a feeling of the degree of separation between these clusters. Notice that the order of the leaves does not necessarily correspond to the order of the labeling of the data. The order of the leaves in `R`'s algorithm is to put the *next cluster split on the right*. This is the rule that we will use in the module, and that you should use for any examined part of this module.

Figure 61 (bottom right) shows the dendrogram for the data in the same Figure (other subplots), using complete linkage. At the first step elements 3 and 4 were merged, and subsequently at the second step elements 2 and 6 were merged. The third step combined element 5 with cluster $\{3, 4\}$, afterwards element 1 was combined with cluster $\{2, 6\}$, and finally all elements were combined in a single cluster). The height of the 'jumps' in the dendrogram give an indication of the number of clusters that might be adequate to describe the data. Big jumps indicate clusters that lie far apart and should not be combined. In our example, the dendrogram suggests the existence of two main clusters, agreeing with the visual impression in Figure 61.

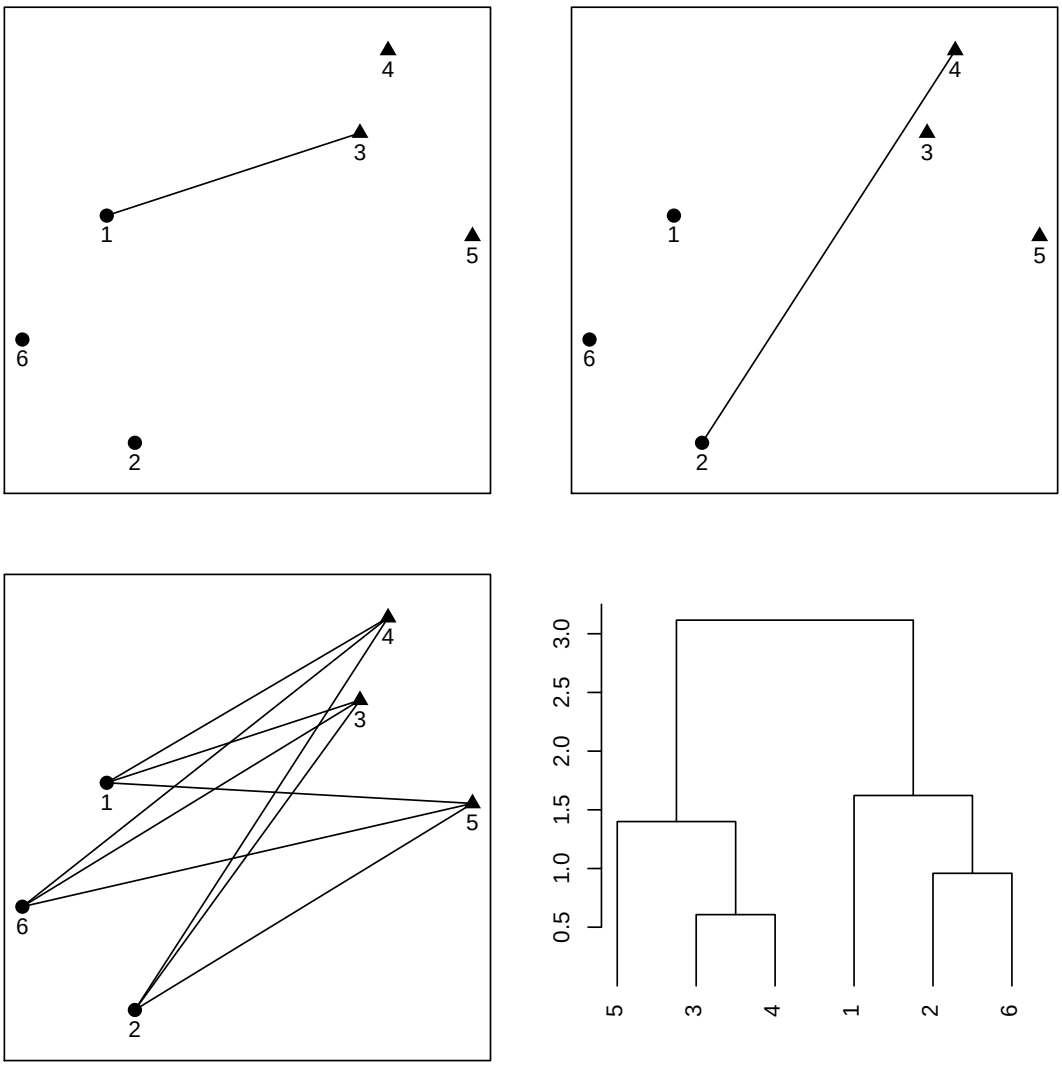



Figure 61: Linkage methods to measure distance between two clusters. Top left: single linkage; top right: complete linkage; Bottom left: average linkage; Bottom right: dendrogram of the 6 data points using complete linkage.

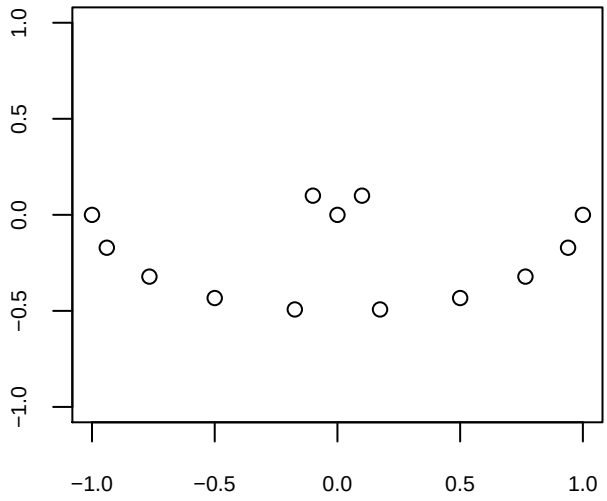



Figure 62: Circle data.

**Example 7.2.1.** Suppose that we wish to cluster the data shown in Figure 62. We consider using single, complete and average linkage. We start by running the hierarchical clustering algorithm with each linkage method (function `hclust` in R) and plotting the dendrogram in each case. The dendrograms for the three linkages are provided in the left panels of Figure 63. When using single linkage (top) the dendrogram suggests 2 main clusters, which are indicated in the corresponding top right panel. Complete and average linkage (middle and bottom, respectively) suggest 3 main clusters, which are indicated in the corresponding right panels.

From the perspective of single linkage, it is fine to have a long chain of concatenated elements, even if some elements in the same cluster end up being quite far from each other. In contrast, both complete and average linkage penalize having two elements that are far away from each other but are assigned to the same cluster. From these results it is hard to state which linkage method is more appropriate in this particular case. The final choice would typically depend on what the clustering results will be used for.

---

**Example 7.2.2.** We now consider hierarchical clustering the Iris data. Figure 64 (bottom subfigure) shows a principal components plot where each flower has been coloured according to its species. We applied hierarchical clustering (R code below) with Euclidean distance and complete linkage, obtaining the dendrogram shown in Figure 64 (top left). Although the dendrogram suggests 2 clusters we divide the sample into 3 clusters, shown in the top right panel. The cluster on the left corresponds to species 1, but the two clusters on the right do not exactly correspond

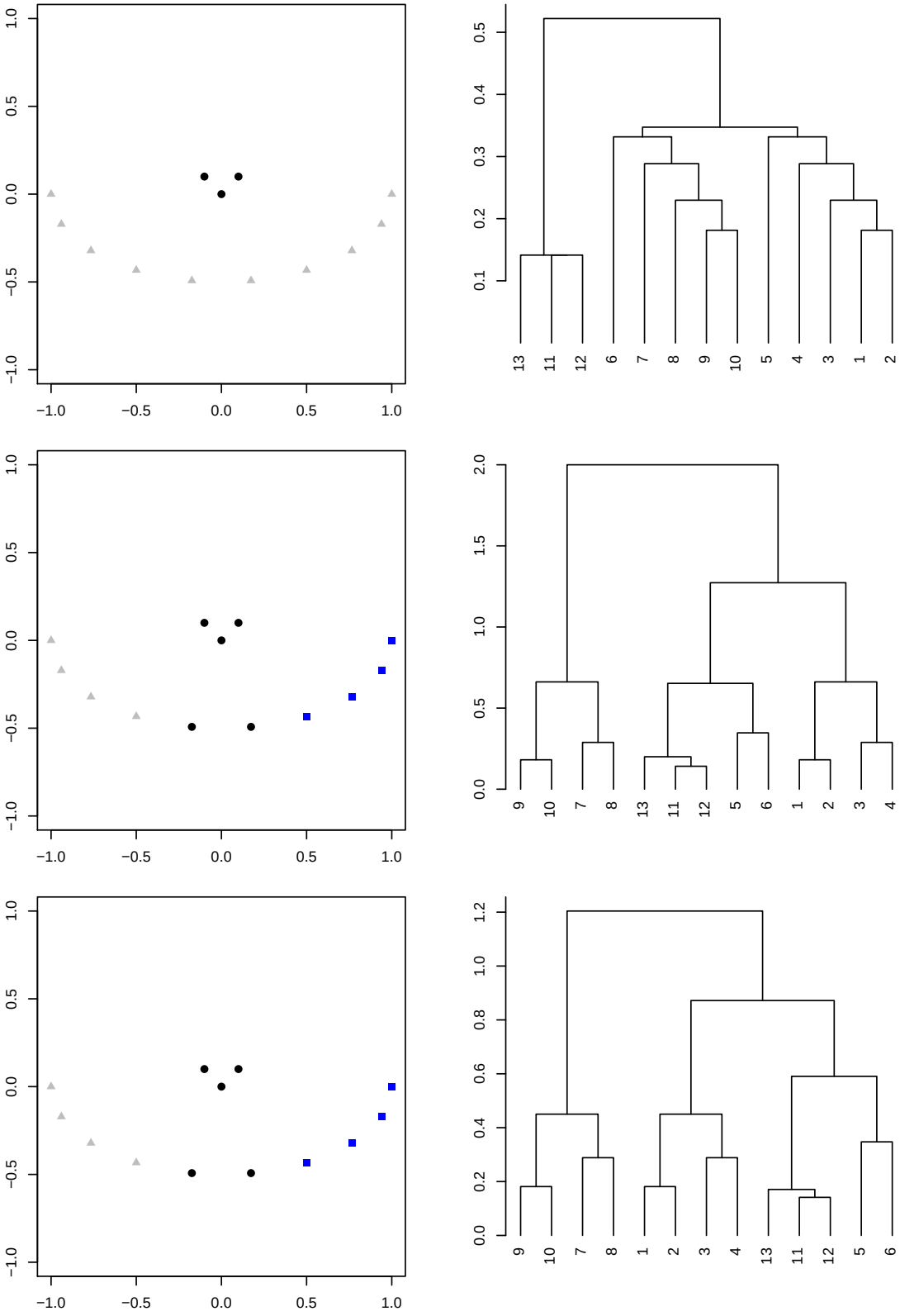



Figure 63: Hierarchical clustering with Euclidean distance. Top: single linkage, Middle: complete linkage, Bottom: average linkage.

to the two remaining species.

Based on the 3 cluster solution, we compute the three within groups covariance matrices and combined them into the pooled covariance matrix $S_p$. We then use $S_p$ to define Mahalanobis distances and apply hierarchical clustering with complete linkage. This is the same algorithm as before, simply changing the definition of distance. The result is shown in Figure 64, bottom panels. The dendrogram now suggests two very clearly separated clusters, one of which is further sub-divided into two clusters. These 3 clusters better match the three species of flowers.

```
data(iris)
pc <- prcomp(iris[,1:4])$x
#col <- as.numeric(iris$Species)
#plot(pc[,1],pc[,2],col=col,pch=col,xlab='PC1',ylab='PC2')

h1 <- hclust(dist(iris[,1:4]), method='complete')
clus <- cutree(h1, k=3)

#Compute pooled within-groups covariance
S1 <- cov(iris[clus==1,1:4])
S2 <- cov(iris[clus==2,1:4])
S3 <- cov(iris[clus==3,1:4])
n1 <- sum(clus==1); n2 <- sum(clus==2); n3 <- sum(clus==3)
Sp <- ((n1-1)*S1 + (n2-1)*S2 + (n3-1)*S3)/(n1+n2+n3-3)

#The code below computes pairwise Mahalanobis distances
require(ICSNP)
pairdiff <- pair.diff(as.matrix(iris[,1:4]))
d <- mahalanobis(pairdiff,center=rep(0,ncol(pairdiff)),cov=Sp,
                 inverted=FALSE)
dmat <- matrix(0,nrow=nrow(iris),ncol=nrow(iris))
dmat[lower.tri(dmat)] <- d; dmat <- dmat + t(dmat)

h2 <- hclust(as.dist(dmat), method='complete')
clus2 <- cutree(h2, k=3)

op <- par(mfrow=c(3,2), mai=c(.05,.33,.3, .1), cex.main=1)
layout(matrix(c(1,3,5, 2,4,5), nrow=3, ncol=2))
plot(h1, main='Euclidean distance, Complete Linkage',xlab='',hang=-1,
     cex=.2)
abline(h=3.8, lty=2)
#
plot(h2,main='Mahalanobis distance, Complete Linkage', xlab='',hang=-1,
     cex=.2)
abline(h=60, lty=2)
#
par(mai=c(.1,.1,.05, .1))
```

```
#
plot(pc[,1],pc[,2],col=clus,pch=clus, xaxt='n', yaxt='n')
#
#plot(pc[,1],pc[,2],col=clus2,pch=clus2,xlab='PC1',ylab='PC2')
plot(pc[,1],pc[,2],col=clus2, pch=clus2, xaxt='n', yaxt='n')
#
par(mai=c(.0,.1,.5, .1))
col <- as.numeric(iris$Species)
plot(pc[,1],pc[,2],col=col,pch=col,xlab='PC1',ylab='PC2',
     main='True Groups (Species)', xaxt='n', yaxt='n', line=0.3)
par(op)
```

---

**Advantages of hierarchical clustering**

- Dendrogram gives an idea of the number of clusters,
- Can cluster both individuals and variables,
- Only required inputs are distances (or similarities).

**Limitations of hierarchical clustering**

- It can be very sensitive to arbitrary choices (linkage method),
- Will not re-allocate elements assigned to the wrong cluster in early iterations,
- It has no interpretation in terms of an underlying probability model.

## 7.3 $K$-Means Clustering

Another popular clustering strategy is the so-called $K$-*means* clustering. The basic idea is to consider a certain number $K$ of clusters, which must be specified in advance, and allocate each observation to the cluster with nearest centroid (mean). Here distance is usually taken to be the Euclidean distance (either using standardized or non-standardized observations), which implies that $K$-means tries to find spherical clusters. Of course, nothing stops us from computing other distances, e.g. Mahalanobis distances which tries to find elliptical clusters.

**Algorithm 7.3.1** ($K$-means clustering with Euclidean distance)**.**

1. Partition the $n$ objects into $K$ initial clusters
2. Compute the centroid (mean) of each cluster
3. Re-allocate each point to the cluster with closest centroid (in Euclidean distance).

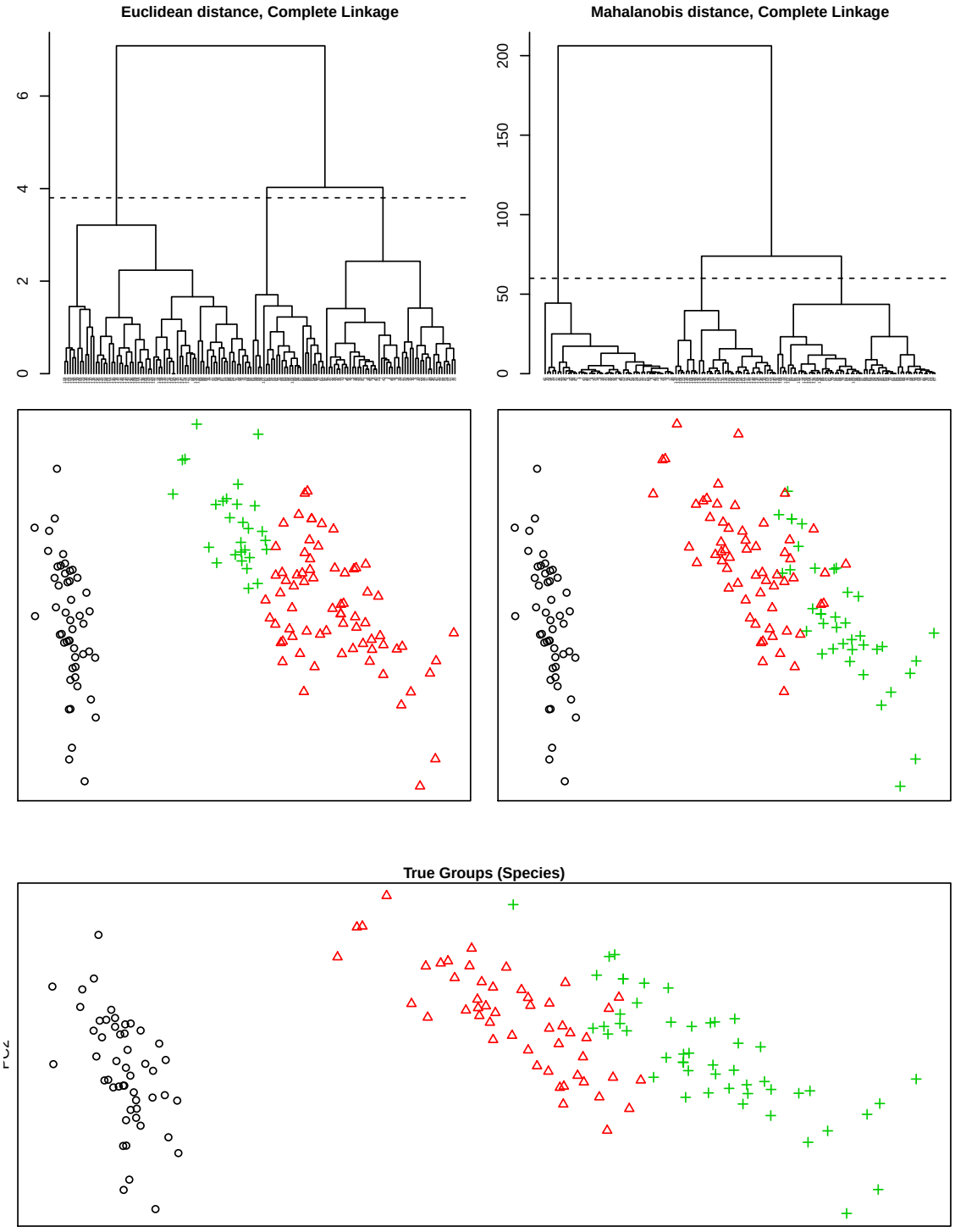



Figure 64: Hierarchical clustering of Anderson's Iris dataset using Euclidean distance (top left) and Mahalanobis distance (top right). The cutoff level to obtain 3 clusters is denoted by a horizontal dashed line. The corresponding clusters are in the middle plots (middle left: Euclidean distance; middle right: Mahalanobis distance). These are plots of the first 2 PC scores, where colour (or shape) indicate the inferred clusters. Bottom: PC plot (PC scores 1 vs. 2) with observations coloured according to the species indicated in the dataset (true underlying groups).

Steps 2-3 are repeated until no more objects are re-allocated.

**Remark 7.3.1 (Variants of $K$-means).**

1. One could use a distance that is not the Euclidean distance. For instance, in Example 7.3.2, a clustering is performed using the Mahalanobis distance.

2. Instead of computing the centroid in step 2., one could choose the most central observation as the representative of each cluster. One could define the most central object as

$$i \triangleq \arg\min_{i \in C_k} \sum_{j \in C_k} d_{ij}^2.$$

This is a generalization of $K$-means. Another variant would be to take the medoids

$$i \triangleq \arg\min_{i \in C_k} \sum_{j \in C_k} d_{ij},$$

as the cluster center. This last version of the algorithm is called $K$-medoids clustering. Note that the last two equations do not require that the objects lie in a Euclidean space, since only the pairwise distances $d_{ij}$ are needed to run the algorithm. Notice however that the computational cost of (a naive implementation of) the $K$-medoids clustering is $O(n_k^2)$ in step 2., instead of $O(n_k)$ for step 2. of $K$-means, where $n_K \triangleq |C_k|$ is the cardinality of cluster $C_k$.

Nice animations and graphics showing the iterations of the $K$-means algorithm can be found online (e.g. `https://en.wikipedia.org/wiki/K-means_clustering`), and some applications are given in `https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf`, sections 14.3.8. and 14.3.9. An important input to the $K$-means algorithm is the choice of $K$. Various heuristic methods exist; we only give one here, sometimes known as the *elbow rule*: for each $K = 1, \ldots, K_{\max}$, compute the within cluster dissimilarities

$$W_K = \sum_{k=1}^{K} \sum_{i,j \in C_k} d^2(x_i, x_j),$$

where $C_1, \ldots, C_K$ are the clusters obtained by $K$-means. Then one plots $K$ versus $W_K$, and looks for an elbow in the figure. The number of clusters suggested by this method is at the elbow of the figure, that is $K^*$ such that $\{W_{K+1} - W_K | K < K^*\} \gg \{W_{K+1} - W_K | K \geq K^*\}$. This is illustrated in Figure 65.

**Example 7.3.2.** We apply $K$-means clustering to Anderson's Iris data, after standardizing the four variables to mean 0 and variance 1. Figure 66 (left) shows the result of the standard $K$-means algorithm based on Euclidean distances. As we observed with hierarchical clustering, the clusters do not perfectly correspond to the

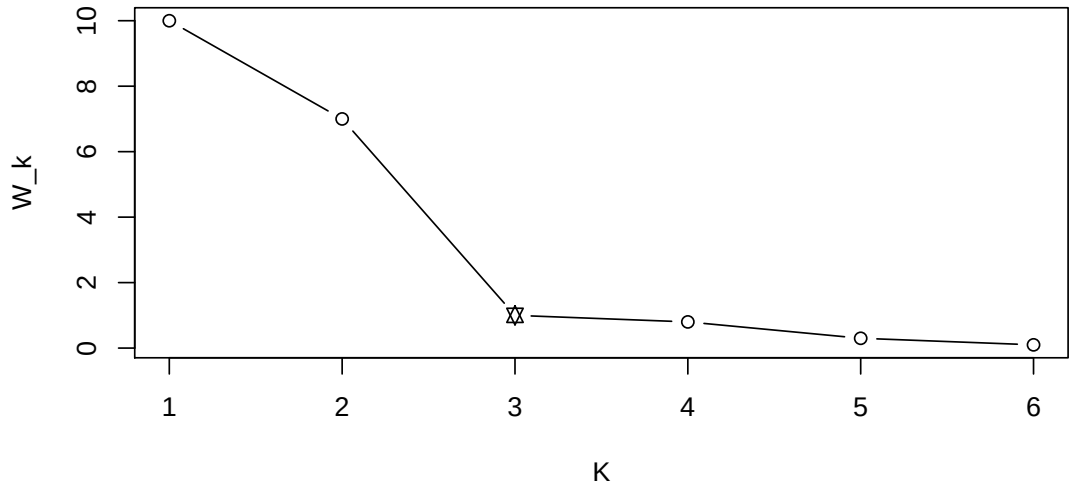



Figure 65: Plot of within cluster dissimilarities for different number of clusters. The elbow criterion tells us to choose $K = 3$ clusters.

species. $K$-means encourages spherical clusters, which result in the points to the right of the plot being divided into two roughly spherical clusters.

Next we used the three obtained clusters to compute the within-groups covariances, which were then combined into a pooled covariance matrix $S$. This matrix was used to compute Mahalanobis distances. Because the option with Mahalanobis distances is not implemented in R, we use the fact that Mahalanobis distances $\mathbf{x}^\mathsf{T} S^{-1}\mathbf{x}$ are equivalent to Euclidean distances $\mathbf{z}^\mathsf{T}\mathbf{z}$ for $\mathbf{z} = S^{-1/2}\mathbf{x}$ (see R code below). Figure 66 (right) shows the $K$-means result with Mahalanobis distances, which resembles the three species quite closely.

```
data(iris)
pc <- prcomp(iris[,1:4])$x
z <- scale(iris[,1:4])

set.seed(1)
op <- par(mfrow=c(1,2), mai=c(.1,.1,.1,.1))
ans <- kmeans(z, centers=3)
clus <- ans$cluster
plot(pc[,1],pc[,2],col=clus,pch=clus,xlab='PC1',ylab='PC2', xaxt='n',
     yaxt='n')
#
S1 <- cov(z[clus==1,]); S2 <- cov(z[clus==2,]); S3 <- cov(z[clus==3,])
n1 <- sum(clus==1); n2 <- sum(clus==2); n3 <- sum(clus==3)
Sp <- ((n1-1)*S1 + (n2-1)*S2 + (n3-1)*S3)/(n1+n2+n3-3)
sqrtSinv <- eigen(Sp)$vectors %*% diag(1/sqrt(eigen(Sp)$values))
```

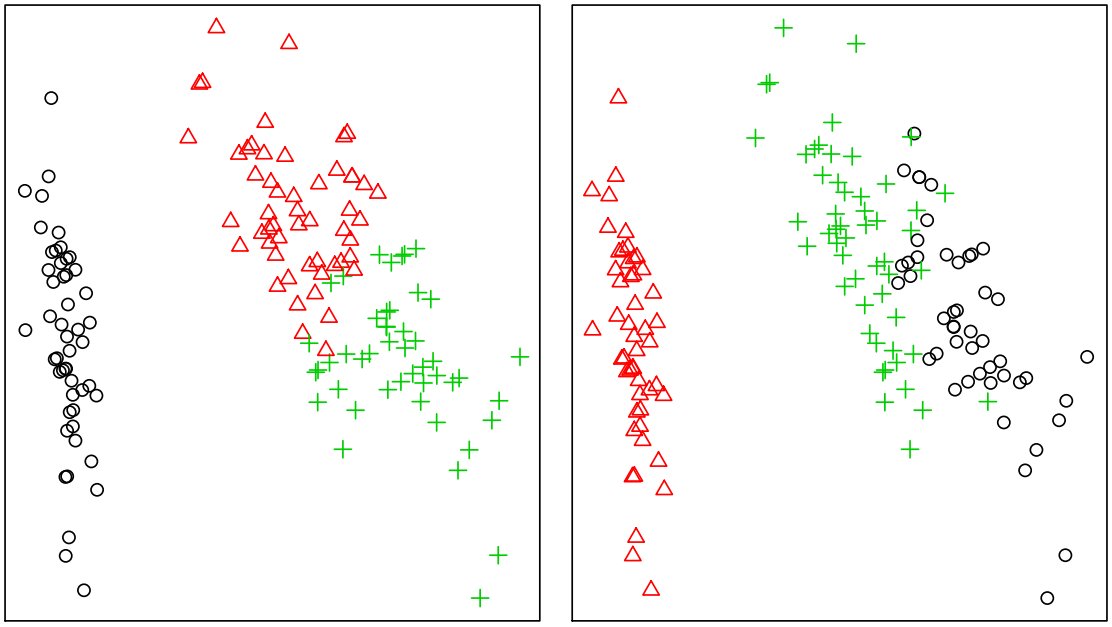


Figure 66: $K$-means clustering of Anderson's Iris data (Left: Euclidean distance; Right: Mahalanobis Distance). Notice that the colour coding of the clusters changes (a clustering algorithm cannot distinguish between labellings).

```
z2 <- z %*% t(sqrtSinv)
#
clus2 <- kmeans(z2, centers=3)$cluster
plot(pc[,1],pc[,2],col=clus2,pch=clus2,xlab='PC1',ylab='PC2',
     xaxt='n', yaxt='n')
par(op)
```

---

**Advantages of $K$-means clustering**

- Flexibility: it re-allocates observations at each iteration,
- No need to specify linkage,
- Has connections with clustering based on statistical models (mixture of multivariate Normal distributions).

**Limitations of $K$-means clustering**

- Need to fix the number of clusters in advance,
- Computing centroids requires providing the co-ordinates—distance matrix alone is not enough.

## 7.4 Other Clustering Algorithms

There are many other algorithms beyond hierarchical and $K$-means clustering. It is beyond the scope of this course to see these more advanced methods, but here we briefly mention a few of them so that you are familiar with their existence.

### 7.4.1 Combination of hierarchical and $K$-means clustering

One option is to combine hierarchical and $K$-means clustering, for instance performing an initial hierarchical clustering followed by $K$-means.

**Advantages of hybrid hierarchical/$K$-means clustering**

- Hierarchical clustering suggests the number of clusters,
- $K$-means flexibly re-allocates individuals,
- Less sensitive to linkage method.

### 7.4.2 Model-based clustering

Another popular approach is to perform probabilistic clustering. Rather than simply assigning individuals to clusters, here we also report the probability that each observation belongs to a given cluster. The approach is based in identifying each cluster with a different probability distribution. For instance, let $\mathbf{X}_i \in \mathbb{R}^p$ be the random variables for individual $i = 1, \ldots, n$. We introduce a fictitious variable $Z_i$ that indicates the cluster that individual $i$ belongs to. This variable is usually called a *latent variable*, as we never get to observe it, it is simply a trick to be able to write a probability model. For instance, let us assume that data in cluster $k = 1, \ldots, K$ follow a multivariate Normal, then the model is formulated as

$$\begin{aligned} \mathbf{X}_i \mid Z_i = k &\sim N_p(\boldsymbol{\mu}_k, \Sigma_k) \\ \mathbb{P}(Z_i = k) &= \pi_k. \end{aligned}$$

The goal is to determine $\mathbb{P}(Z_i = k \mid \mathbf{X}_i)$, the cluster probabilities for individidual $i$. With these in hand we can assign each individual to the most probable cluster. Importantly, the probabilities also tell us how certain we are of the cluster assignments.

By the law of total probability the density of $\mathbf{X}_i$ is

$$f(\mathbf{X}_i) = \sum_{k=1}^{K} \pi_k N(\mathbf{X}_i; \boldsymbol{\mu}_k, \Sigma_k). \tag{7.4.1}$$

Therefore we can define the likelihood $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} f(\mathbf{x}_i)$ as usual, which we can maximize with respect to $(\pi_k, \boldsymbol{\mu}_k, \Sigma_k)$, $k = 1, \ldots, K$ to obtain maximum likelihood estimates (or posterior distributions, in a Bayesian setup). Implementations

of this method can be found in R packages such as `mclust`, `EMCluster` or `bayesm`. By Bayes' theorem, the cluster probabilities are given by

$$\mathbb{P}(Z_i = k \mid \mathbf{X}_i, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \boldsymbol{\pi}) = \frac{N(\mathbf{X}_i; \boldsymbol{\mu}_k, \Sigma_k)\pi_k}{\sum_{j=1}^{K} N(\mathbf{X}_i; \boldsymbol{\mu}_j, \Sigma_j)\pi_j}, \tag{7.4.2}$$

which can be estimated for instance by plugging in the MLEs for $(\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)$, $k = 1, \ldots, K$.

In fact, the $K$-means algorithm with Euclidean distance is strongly connected with model (7.4.1) when $\Sigma_k$ are spherical matrices, and $K$-means with Mahalanobis distances to the case with general $\Sigma_k$.

**Advantages of model-based clustering**

- Cluster probabilities measure uncertainty in cluster allocations
- The number of clusters $K$ is a parameter in the model, and can be selected with standard model choice criteria (AIC, BIC etc.)
- No need to specify linkage
- Model automatically defines the relevant measure of distance
- Model assumptions can be checked

We briefly review two popular model choice criteria, AIC and BIC, which are useful when comparing models with different number of parameters. We cannot compare such models using only the likelihood evaluated at the maximum likelihood estimate for the parameters, because the likelihood always favours the more complex model. Instead, we need to find a trade-off between the fit to the data (measured by the likelihood) and model complexity. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be $n$ independent observations from the density $f(\mathbf{x} \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the parameters of the probability distribution. Let $q = \dim(\boldsymbol{\theta})$ be the number of parameters in the model and $\hat{\boldsymbol{\theta}}$ their maximum likelihood estimate, that is

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{n} f(\mathbf{x}_i \mid \boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}).$$

**Definition 7.4.1.** Akaike's information criterion is defined as

$$\mathrm{AIC} = -2 \log L(\hat{\boldsymbol{\theta}}) + 2q.$$

Similary, the Bayesian information criterion is

$$\mathrm{BIC} = -2 \log L(\hat{\boldsymbol{\theta}}) + q \log n,$$

where $q$ is the number of parameters and $n$ the sample size.

The first term in both criteria is proportional to minus the log-likelihood evaluated at the MLE. Small values correspond to high likelihood, which indicates a good fit. The second term is a penalization for the number of parameters, so that complex models with large $q$ are penalized. Small AIC or BIC values indicate models that achieve a reasonable fit with relatively few parameters, and hence are useful when comparing models of different dimensionality.

**Example 7.4.2.** Suppose we wish to compare the two following linear regression models

$$\begin{aligned} M_1 : Y_i &= \beta_0 + \beta_1 X_{1i} + E_i \\ M_2 : Y_i &= \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + E_i, \end{aligned}$$

where $E_i \sim N(0, \sigma^2)$ are independent for $i = 1, \ldots, n$. Because model 1 is a particular case of model 2, the likelihood under model 2 evaluated at the MLE will always be larger than that for model 1.

Define by $\hat{\theta}^{(1)} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\sigma})$ the MLE under model 1 and by $\hat{\theta}^{(2)} = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\sigma})$ that under model 2. The respective AIC values are

$$\begin{aligned} \mathrm{AIC}_1 &= -2 \log L(\hat{\boldsymbol{\theta}}^{(1)}) + 6 \\ \mathrm{AIC}_2 &= -2 \log L(\hat{\boldsymbol{\theta}}^{(2)}) + 8, \end{aligned}$$

so that we choose model 1 when $\log L(\hat{\boldsymbol{\theta}}^{(2)}) - \log L(\hat{\boldsymbol{\theta}}^{(1)}) < 1$. Similary, using BIC we choose model 1 when $\log L(\hat{\boldsymbol{\theta}}^{(2)}) - \log L(\hat{\boldsymbol{\theta}}^{(1)}) < \frac{1}{2} \log n$.

Both for AIC and BIC, we only choose model 2 when the difference in log-likelihoods is deemed large enough.

---

**Example 7.4.3.** Consider now the case of model-based clustering, where each cluster is associated with a multivariate Normal component. The likelihood for a model with $K$ clusters is

$$L_K = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k),$$

where $\mathbf{x}_i \in \mathbb{R}^p$. Let $\hat{L}_K$ be the likelihood evaluated at the MLE $(\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi_k})$ for $k = 1, \ldots, K$. The number of parameters is $q_K = K(p + p(p+1)/2) + K - 1$, where we consider that there are only $K - 1$ independent $\pi_k$'s (since $\pi_K = 1 - \sum_{k=1}^{K-1} \pi_k$). Therefore the AIC and BIC for a model with $K$ clusters are

$$\begin{aligned} \mathrm{AIC}_K &= -2 \log(\hat{L}_K) + 2 q_K \\ \mathrm{BIC}_K &= -2 \log(\hat{L}_K) + q_K \log(n). \end{aligned} \tag{7.4.3}$$

Fortunately, both finding MLEs and computing the BICs are implemented in the R package `mclust`.

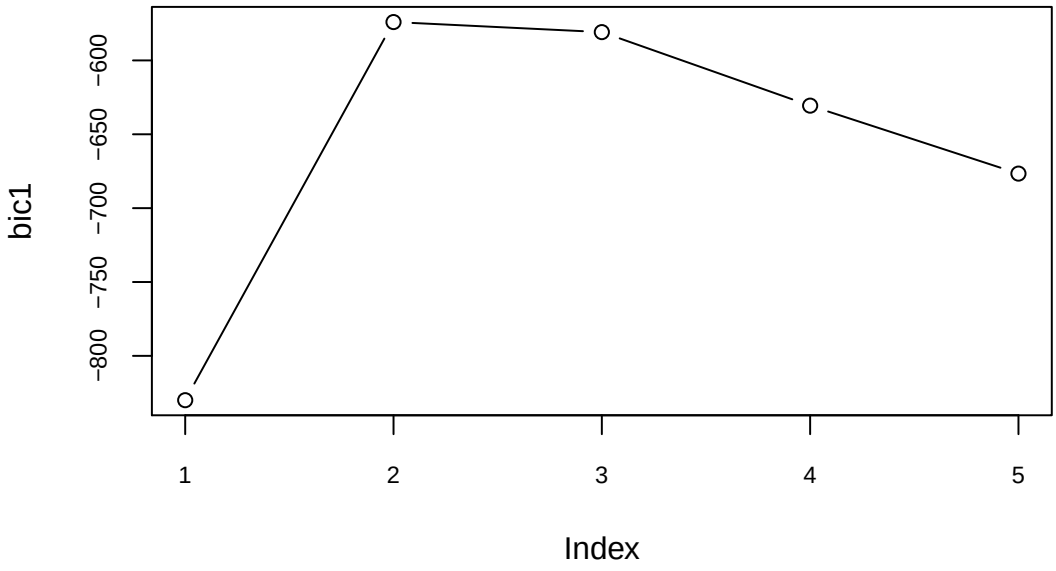



Figure 67: BIC values for `iris` dataset.

```
library(mclust)
library(mvtnorm)
data(iris)

bic1 <- mclustBIC(iris[,1:4],G=1:5,modelNames='VVV', verbose=F)
mclust1 <- mclustModel(iris[,1:4],BICvalues=bic1)
pro <- mclust1$par$pro
m <- mclust1$par$mean
v <- mclust1$par$variance$sigma
clusprob <- mclust1$z
clus <- ifelse(clusprob[,1]>.5,1,2)
op <- par(cex.axis=.75)
plot.default(bic1, type='b', col=1)
par(op)
```

Let us see what results we get on the Iris dataset. Figure 67 shows the negative of the BIC for $K = 1, \ldots, 5$ clusters. The best number of clusters according to this criterion is $K = 2$, followed closeby by $K = 3$. Clearly, $K = 1$ provides a very bad fit and $K = 4, 5$ add model complexity without important gains in model fit.

Figure 68 (left) shows the first two principal components, along with the contours of the $K = 2$ multivariate Normal components. Observations are coloured according to the component (cluster) with highest probability. The right panel shows the contours for the model with $K = 3$ components, which we can see closely resembles

the 3 species of flowers actually present in the data.

```
library(mclust)
library(mvtnorm)
data(iris)

x <- as.matrix(iris[,1:4])
e <- eigen(cov(x))$vectors
pcs <- x %*% e[,1:2]
xlim <- c(min(pcs[,1]),max(pcs[,1]))
ylim <- c(min(pcs[,2]),max(pcs[,2]))

xseq <- seq(min(pcs[,1]),max(pcs[,1]),length=300)
yseq <- seq(min(pcs[,2]),max(pcs[,2]),length=300)
xgrid <- expand.grid(xseq,yseq)

## 2 clusters
m1 <- t(e[,1:2]) %*% m[,1,drop=FALSE]
S1 <- t(e[,1:2]) %*% v[,,1] %*% e[,1:2]
f1 <- dmvnorm(xgrid, m1, S1)
m2 <- t(e[,1:2]) %*% m[,2,drop=FALSE]
S2 <- t(e[,1:2]) %*% v[,,2] %*% e[,1:2]
f2 <- dmvnorm(xgrid, m2, S2)
#
op <- par(mfrow=c(1,2), mai=c(1,1,1,1)/8)
plot(pcs, xlab='PC1',ylab='PC2',xlim=xlim,ylim=ylim,cex.lab=1.25,
     pch=clus,col=clus, xaxt='n', yaxt='n')
par(new=TRUE)
contour(x=xseq, y=xseq, z=matrix(f1,nrow=length(xseq),ncol=length(xseq)),
        xlab='',ylab='',xaxt='n',yaxt='n',drawlabels=FALSE,col=1)
par(new=TRUE)
contour(x=xseq, y=xseq, z=matrix(f2,nrow=length(xseq),ncol=length(xseq)),
        xlab='',ylab='',xaxt='n',yaxt='n',drawlabels=FALSE,col=2)

## 3 clusters
bic1 <- mclustBIC(iris[,1:4],G=3,modelNames='VVV', verbose=F)
mclust1 <- mclustModel(iris[,1:4],BICvalues=bic1)
pro <- mclust1$par$pro
m <- mclust1$par$mean
v <- mclust1$par$variance$sigma
clusprob <- mclust1$z
clus <- apply(clusprob,1,which.max)
#
m1 <- t(e[,1:2]) %*% m[,1,drop=FALSE]
S1 <- t(e[,1:2]) %*% v[,,1] %*% e[,1:2]
f1 <- dmvnorm(xgrid, m1, S1)
```

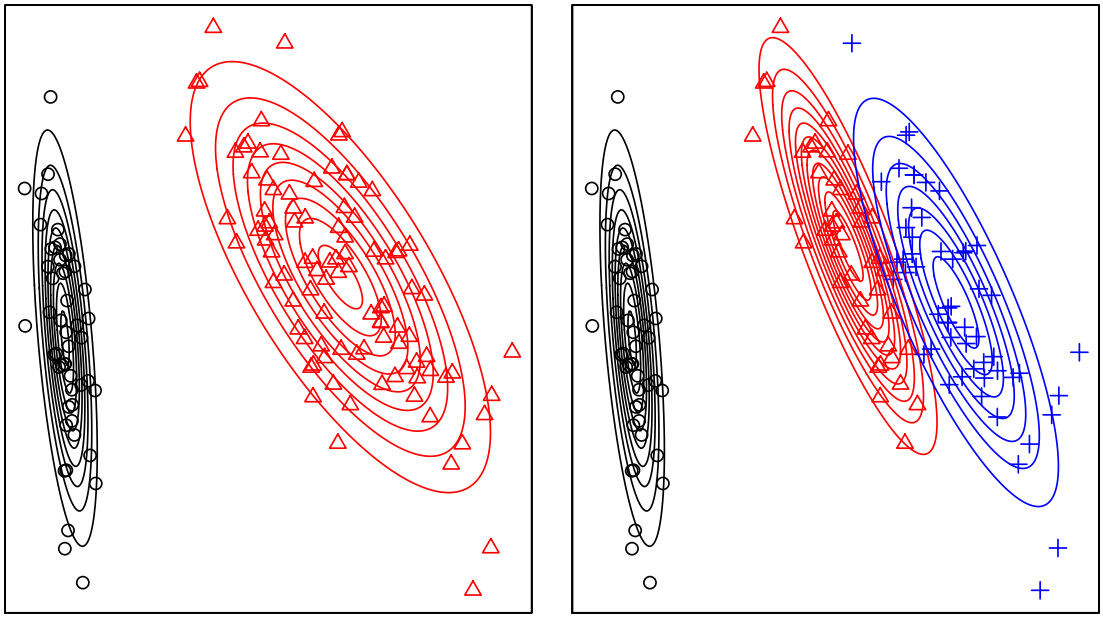


Figure 68: Multivariate Normal contours for the 2 cluster (left) and 3 cluster (right) solutions. Observations are coloured according to the cluster with highest probability

```
m2 <- t(e[,1:2]) %*% m[,2,drop=FALSE]
S2 <- t(e[,1:2]) %*% v[,,2] %*% e[,1:2]
f2 <- dmvnorm(xgrid, m2, S2)
m3 <- t(e[,1:2]) %*% m[,3,drop=FALSE]
S3 <- t(e[,1:2]) %*% v[,,3] %*% e[,1:2]
f3 <- dmvnorm(xgrid, m3, S3)

col <- clus; col[col==3] <- 4
plot(pcs, xlab='PC1',ylab='PC2',xlim=xlim,ylim=ylim,cex.lab=1.25,
     pch=clus,col=col, xaxt='n', yaxt='n')
par(new=TRUE)
contour(x=xseq, y=xseq, z=matrix(f1,nrow=length(xseq),ncol=length(xseq)),
        xlab='',ylab='',xaxt='n',yaxt='n',drawlabels=FALSE,col=1)
par(new=TRUE)
contour(x=xseq, y=xseq, z=matrix(f2,nrow=length(xseq),ncol=length(xseq)),
        xlab='',ylab='',xaxt='n',yaxt='n',drawlabels=FALSE,col=2)
par(new=TRUE)
contour(x=xseq, y=xseq, z=matrix(f3,nrow=length(xseq),ncol=length(xseq)),
        xlab='',ylab='',xaxt='n',yaxt='n',drawlabels=FALSE,col=4)

par(op)
```

## 7.5 Cluster stability

Given any input data, any clustering algorithm will report potential clusters. It is therefore important to assess how reliable, or reproducible, these clusters are.

- It is wise to try multiple clustering algorithms and compare the solutions.
- For algorithms like $K$-means with an element of randomness (due to choice of initial clustering in $K$-means) it is useful to run the algorithm multiple times.

While useful, these strategies only assess robustness of clustering results in our observed data. Usually we really want to check that these clusters would also be observed in an *independent* data set. If we have lots of data, one option is to split them into two data sets, run separate algorithms on each data and compare the solutions.

As the amount of data is usually limited, a popular option is to generate a new dataset that is a perturbed version of the original data. For instance, we could add some random noise to the observed data, in which case we would have to decide what amount of noise is reasonable. Another possibility is to obtain bootstrap samples (i.e. sample $n$ individuals with replacement from the original data, possible adding random noise) and re-apply the algorithm.

**Algorithm 7.5.1** (Assessing cluster reproducibility)**.**
For $b = 1, \ldots, B$, do the following steps:

(a) Create a perturbed data matrix $X^{(b)}$ from the original data $X$ (e.g. bootstrapping or adding random noise),

(b) Apply the clustering algorithm to $X^{(b)}$,

(c) Record the cluster characteristics (centroids, number of clusters etc.) and cluster assignments.

After $B$ iterations, report the distribution of the cluster characteristics (e.g. centroids, number of clusters for a given distance threshold) or the proportion of times two individuals were clustered together.

Because at each iteration we may obtain different clustering results, studying how stable the clustering characteristics are gives us a sense of the uncertainty regarding the clusters detected in the original data $X$.

**Example 7.5.1.** We generated $n_1 = 30$ independent draws from $N_{100}(\boldsymbol{\mu}_1, I)$, where $I$ is the $100 \times 100$ identity matrix and $\mu_{1j} \sim N(0, \sigma = 0.5)$ independently for $j = 1, \ldots, 100$. We then generated $n_2 = 20$ additional samples from $N_{100}(\boldsymbol{\mu}_2, I)$, where $\mu_{2j} \sim N(0, \sigma = 0.5)$ (and are hence different from the $\mu_{1j}$s). That is, we generated data from 2 spherical clusters that differ only in their means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$.

We merged all data and applied hierarchical clustering with Pearson correlation distance and complete linkage. Figure 69 (top) shows the dendrogram. While not overwhelmingly clear, the dendrogram suggests the existence of 2-3 clusters.

To better assess cluster reproducibility, we obtained 500 bootstrap samples. For each bootstrap sample we repeated the hierarchical clustering and divided the individuals into the 2 main clusters. We then compared if individuals that were clustered together in the original data were again assigned to the same cluster in the bootstrap samples, and computed a percentage of agreement (0 if the samples never clustered together, 100 if they always clustered together). The middle panel in Figure 69 shows the agreement scores. Yellow corresponds to 100 (individuals always clustered together) and blue to 0 (individuals always in different clusters). The plot reveals that individuals that were in the larger cluster in the original data also tended to be in the same cluster in the bootstrap samples. Similarly, the smaller cluster also showed high agreement scores.

Next we tried defining 4 clusters on these same data. We again repeated the bootstrap analysis and computed agreement scores (Figure 69, bottom). The four clusters were clearly not reproducible in the bootstrap samples. The sub-clusters obtained when dividing each of the main two clusters are not distinguishable: individuals in two different sub-clusters clustered together very frequently.

Below is the R code used generate the example. It uses the R package `ClassDiscovery`, which can be obtained at `bioinformatics.mdanderson.org/Software/OOMPA`. As a remark, this package assumes that variables are in rows and individuals in columns, which is why we transpose the data when calling its functions.

```
library(mvtnorm)
library(ClassDiscovery)
library(fields)
n1 <- 30; n2 <- 20; p <- 100
set.seed(2)
mu1 <- rnorm(p,sd=0.5); mu2 <- rnorm(p,sd=0.5)
x1 <- rmvnorm(n1, mu1, diag(p))
x2 <- rmvnorm(n2, mu2, diag(p))
x <- rbind(x1, x2)

hc <- hclust(distanceMatrix(t(x), 'pearson'), method='complete')
bc <- BootstrapClusterTest(t(x), cutHclust, nTimes=500, k=2,
                           metric='pearson', verbose=FALSE)
bc4 <- BootstrapClusterTest(t(x), cutHclust, nTimes=500, k=4,
                            metric='pearson', verbose=FALSE)

plot(hc,hang=-1,main='',xlab='')
cols <- blueyellow(64)
breaks <- seq(0,1, len=65)
image(bc, dendrogram=hc, col=cols, breaks=breaks, margins=c(2,8))
image.plot(legend.only=TRUE, zlim=c(0,1), col=cols, breaks=breaks)
image(bc4, dendrogram=hc, col=cols, breaks=breaks, margins=c(2,8))
image.plot(legend.only=TRUE, zlim=c(0,1), col=cols, breaks=breaks)
```

To illustrate what happens when no clusters are present in the data at all, we generated 50 draws from $N_{100}(\boldsymbol{\mu}_1, I)$ and applied the same analyses. Figure 70

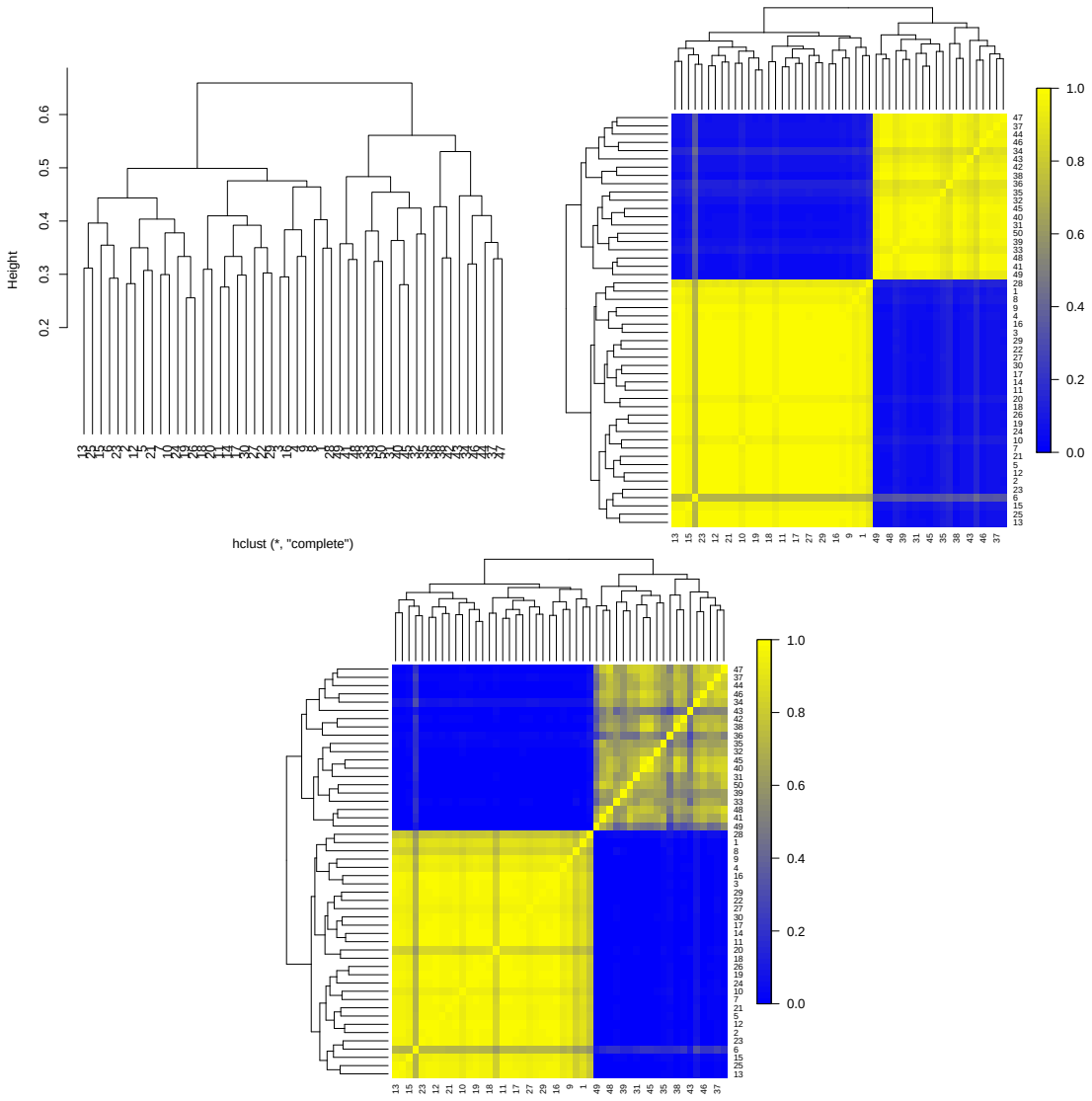



Figure 69: Hierarchical clustering 30 draws from $N_{100}(\boldsymbol{\mu}_1, I)$ and 20 draws from $N_{100}(\boldsymbol{\mu}_2, I)$ (Pearson distance, complete linkage). Dendrogram (top left), Bootstrap agreement for 2 clusters (top right) and 4 clusters (bottom)

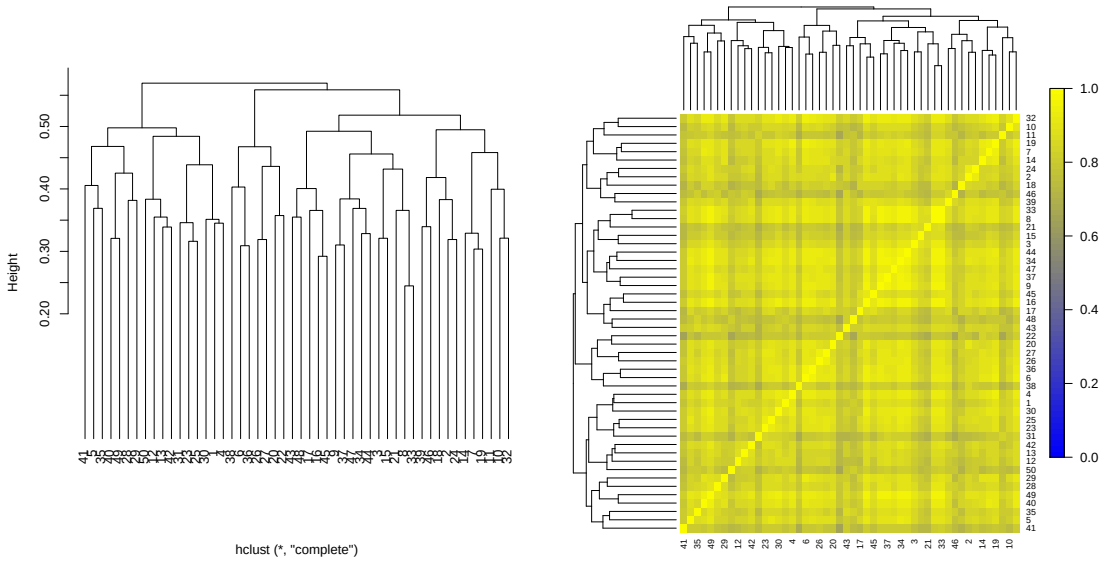



Figure 70: Hierarchical clustering 50 draws from $N_{100}(\boldsymbol{\mu}_1, I)$ (Pearson distance, complete linkage). Top: dendrogram; Bottom: bootstrap agreement for 2 clusters

displays the results. It is hard to get a clear picture from the dendrogram. There appear to be two main branches, although these are not very clearly defined. The situation is much clearer when we look at the bootstrap agreement scores. It is now obvious that individuals from cluster 1 are frequently grouped with individuals from cluster 2 in the bootstrap iterations (and vice versa), which indicates that the 2 main clusters are not reproducible. R code is provided below.

```
library(mvtnorm)
n1 <- 30; n2 <- 20; p <- 100
set.seed(1)
mu1 <- rnorm(p,sd=0.5);
xx <- rmvnorm(n1+n2, mu1, diag(p))
hct <- hclust(distanceMatrix(t(xx),'pearson'),method='complete')
bct <- BootstrapClusterTest(t(xx), cutHclust, nTimes=500, k=2,
                            metric='pearson', verbose=FALSE)

plot(hct,hang=-1,main='',xlab='')
cols <- blueyellow(64)
breaks <- seq(0,1, len=65)
image(bct, dendrogram=hct, col=cols, breaks=breaks, margins=c(2,8))
image.plot(legend.only=TRUE, zlim=c(0,1), col=cols, breaks=breaks)
par(op)
```

## 7.6 Heatmaps and clustering to visualize big data

Clustering is a common strategy used in large data sets to help detect or display certain patterns. For instance, suppose we have a few hundred patients and that for each of them we record tens of thousands of characteristics such as *gene expression*. Clustering can help get a first idea of which patients are similar to each other.

As another example, suppose that after a certain statistical analysis we found a few hundred genes that help distinguish two patient subgroups, say healthy vs. sick individuals. As the following example illustrates, it can then be helpful to cluster both rows (patients) and columns (genes) in our data matrix and produce a *heatmap* (in which colours represent numerical values, for example ranging from dark blue for the lowest value through grey-green for middle values to bright yellow for the highest values).

**Example 7.6.1.** The dataset ALL in R package `ALL` contains gene expression measurements for 12,625 genes and 47 patients with Acute Lymphoblastic Leukemia. Among these 47 patients, 37 showed a certain mutation in gene BCR/ABL (group 1) and the remaining 10 showed a mutation in gene ALL1/AF4 (group 2).

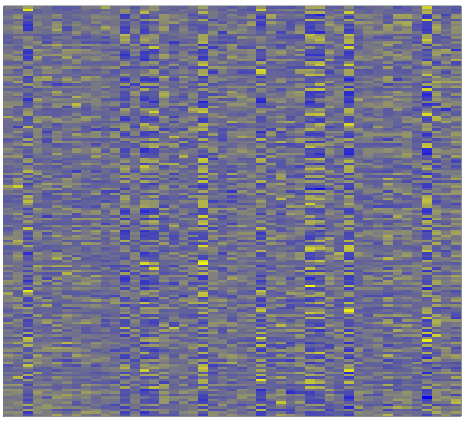
The researchers were interested in finding genes that could help explain the differences between these 2 groups. To this purpose, they performed a certain statistical analysis (which we will not address here) and selected 165 genes that showed strong differences between groups.

Rather than showing a large $165 \times 47$ table, the researchers decided to cluster patients using hierarchical clustering with Euclidean distance and complete linkage. They also clustered genes, computing the distance between 2 genes as $0.5(1 - r_{ij})$, where $r_{ij}$ is the Pearson correlation. They then produced a heatmap, where the intensity of the colours indicates the level of expression for each gene and patient (blue for low, yellow for high expression). Figure 71 shows the result. Here genes are shown in rows and patients in columns, as this is the convention in gene expression studies. We can appreciate two clear groups of genes and patients, where the patient clusters coincide with the original BRL/ABL and ALL1/AF4 subgroups. We observe a distinct expression pattern between subgroups, which is not surprising. The statistical analysis selected genes showing differences between groups, so we expect to see these differences in the heatmap. The point here is that the heatmap helps convey the results of the analysis in a visual and easily interpretable manner that can be conveyed to non-statisticians.

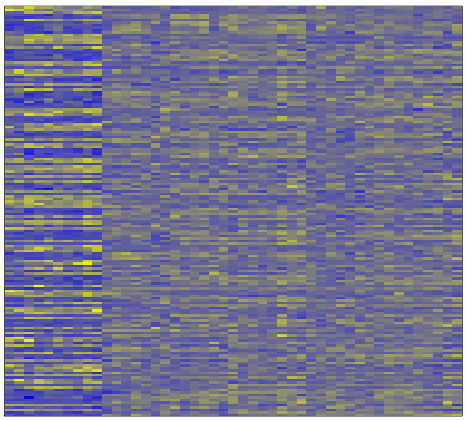
```
library(ALL)
library(limma)
library(ClassDiscovery)

data("ALL")
eset <- ALL[, ALL$mol.biol %in% c("BCR/ABL", "ALL1/AF4")]
f <- factor(as.character(eset$mol.biol))
design <- model.matrix(~f)
fit <- eBayes(lmFit(eset,design))
selected  <- p.adjust(fit$p.value[, 2]) <0.05
```

```
esetSel <- eset [selected, ]
color.map <- function(mol.biol) { if (mol.biol=="ALL1/AF4") "#FF0000"
else "#0000FF" }
patientcolors <- unlist(lapply(esetSel$mol.bio, color.map))

d <- .5*(1-cov2cor(cov(t(exprs(esetSel)))))
h1 <- hclust(as.dist(d), method='complete')

exprs(esetSel) %>% str

##  num [1:165, 1:47] 7.62 8.06 6.88 8.65 6.12 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:165] "1007_s_at" "1039_s_at" "1126_s_at" "1134_at" ...
##   ..$ : chr [1:47] "01005" "03002" "04006" "08001" ...

h2 <- hclust(dist(exprs(esetSel) %>% t), method='complete')
X <- exprs(esetSel); X <- t(scale(t(X)))
image(X %>% t, col=blueyellow(100), xaxt='n', yaxt='n')
image(X[ , h2$order] %>% t, col=blueyellow(100), xaxt='n', yaxt='n')
image(X[ h1$order, h2$order] %>% t, col=blueyellow(100), xaxt='n', yaxt='n')
heatmap(exprs(esetSel), col=blueyellow(100), ColSideColors=patientcolors,
        Rowv=as.dendrogram(h1), cexRow=.3)
```

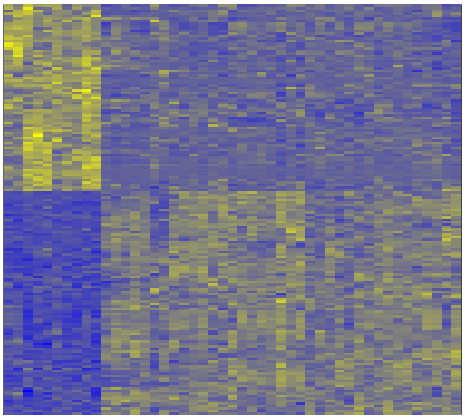
---



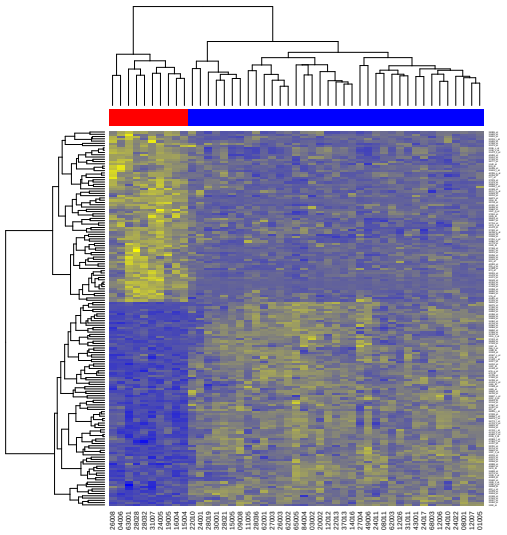
(a) Original data (no reordering of rows or columns)



(b) Data with re-ordered columns



(c) Data with re-ordered rows and columns



(d) Data with re-ordered rows and columns, with the row's and column's dendrograms (obtained using the `heatmap` function)

Figure 71: Heatmap for ALL data, without/with various orderings of the columns (patients) and rows (Genes). The hierarchical clustering is done with complete linkage and correlation distance was used for genes (rows), and Euclidean distance for patients (columns).

# 8 Additional Topic: Kernel Methods (still ST323)

Kernel methods are a broad family of statistical techinques, which extend many well-known methods to work with complex data. In much of this module, we have (sometimes implicitly) exploited linear structure in data, but for some data this is not directly possible and our favourite statistical analyses fail. For example, if data is far from being normally distributed, then it cannot be characterised simply by a mean and covariance matrix.

The main idea of kernel methods is to map our data to a new (usually higher dimensional) vector space, where the data does have some linear structure. Once we have done this, we can then use our earlier techniques more successfully. Instead of working with the observations $(\mathbf{x}_i)_i$ directly, we work with a nonlinear transformation of the observations, that is we work with $\mathbf{y}_i \triangleq \Phi(\mathbf{x}_i), i = 1, \ldots, n$, where $\Phi : \mathbb{R}^p \to \mathcal{H}$ is called the **feature map**, and is typically nonlinear. Here $\mathcal{H}$ may be $\mathbb{R}^q$ with $q \gg p$, but often it is taken to be an infinite dimensional function space.

We focus on the familiar setting where the $\mathbf{x}_i \in \mathbb{R}^p$, but these techniques can be used to analyse much more complicated data types, including genomic data, graphs, text, and images. All we need is a good choice of feature map $\Phi$.

## 8.1 Kernel Principal Component Analysis

Kernel PCA (KPCA) is a tool for exploratory data analysis that extends PCA. It can be useful in cases where the data has some structure that is not linear. For instance Figure 72 shows data coming from two groups. The two groups are not linearly separable, even after performing a PCA, but a KPCA with a Gaussian kernel (explained below) does separate the two groups linearly.

As in PCA, the goal of KPCA is to summarise each datapoint $\mathbf{x}_i \in \mathbb{R}^p$ into uncorrelated scores $s_{ik}, k = 1, \ldots, K$, where $K \leq p$. In standard PCA working with $\mathtt{X}$ directly, Exercise 4.8 tells us that the PC scores are given by the spectral decomposition of $\tilde{\mathtt{X}}\tilde{\mathtt{X}}^\mathsf{T}$, where $\tilde{\mathtt{X}}$ is the column-centered version of $\mathtt{X}$. We know that $\tilde{\mathtt{X}} = H\mathtt{X}$, where $H \triangleq I - \mathbf{1}\mathbf{1}^\mathsf{T}/n$ is the $n \times n$ matrix with $1 - 1/n$ in the diagonal, and $-1/n$ as off-diagonal entries. This tells us therefore that if

$$\tilde{\mathtt{X}}\tilde{\mathtt{X}}^\mathsf{T} = H\mathtt{X}\mathtt{X}^\mathsf{T}H = U\Lambda U^\mathsf{T} \tag{8.1.1}$$

is the spectral decomposition of $H\mathtt{X}\mathtt{X}^\mathsf{T}H$, then the $k$th PC scores $(s_{ik})_i$ are given by the $k$th column of $U\Lambda^{1/2}$. The matrix $\mathtt{X}\mathtt{X}^\mathsf{T}$ has $(i,j)$th entry given by $\mathbf{x}_i^\mathsf{T}\mathbf{x}_j$. In KPCA we replace this inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$ by the inner product between $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ in the vector space $\mathcal{H}$. Another way of thinking about this bypasses the choice of $\Phi$, which can be difficult. Instead, we can directly specify a **kernel function**: $k(\cdot, \cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, which is thought of as

$$k(\mathbf{x}, \mathbf{x}') \triangleq \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}. \tag{8.1.2}$$

**Definition 8.1.1 (Kernel function).**
A function

$$k(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$$

is called a **kernel function** if for all $m \geq 1$ and all $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^p$, the $m \times m$ matrix $K$ with $(K)_{ij} \triangleq k(\mathbf{x}_i, \mathbf{x}_j)$ is symmetric positive semi-definite.

The following two results tell us that defining a kernel function is equivalent to defining the feature map $\Phi$.

**Proposition 8.1.2 (Every feature map defines a kernel).**
If $k(\cdot,\cdot)$ is defined by (8.1.2), then it is a kernel function.

*Proof.* Left as an exercise. □

**Theorem 8.1.3 (Moore–Aronszajn theorem: every kernel has an associated feature map).**
If $k(\cdot,\cdot)$ is a kernel function, then there exists a Hilbert space $\mathcal{H}$ with inner-product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$, and a mapping $\Phi : \mathbb{R}^p \to \mathcal{H}$ such that

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p.$$

*Proof.* The proof uses advanced analysis, and is not examinable. □

We now know that we can work with a kernel instead of using the feature map. Here are examples of well-known kernels :

**linear:** $k(\mathbf{x}, \mathbf{x}') \triangleq \mathbf{x}^\mathsf{T}\mathbf{x}'$,

**polynomial:** $k(\mathbf{x}, \mathbf{x}') \triangleq (1 + \mathbf{x}^\mathsf{T}\mathbf{x}')^d$, for some $d \in \{1, 2, \ldots\}$,

**Gaussian:** $k(\mathbf{x}, \mathbf{x}') \triangleq \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2\sigma^2}\right)$, where $\sigma > 0$.

In particular, the linear kernel is equivalent to taking the feature map to be the identity on $\mathbb{R}^p$.

Let us now turn again to our PC scores computed from the $\mathbf{y}_i$s. We know that the first step is to compute the matrix $H\mathtt{Y}\mathtt{Y}^\mathsf{T}H$, and then compute its spectral decomposition. Using the kernel $k(\cdot,\cdot)$ defined by (8.1.2), we replace $(\mathtt{Y}\mathtt{Y}^\mathsf{T})_{ij}$ by

$$\langle \mathbf{y}_i, \mathbf{y}_j \rangle_{\mathcal{H}} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}} = k(\mathbf{x}_i, \mathbf{x}_j).$$

This leads to the following algorithm for KPCA:

**Algorithm 8.1.1** (Kernel PCA)**.**
Given data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and a kernel function $k(\cdot,\cdot)$,

1. Compute the $n \times n$ matrix $K$, where $(K)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.
2. Compute the row and column centered version of $K$, that is $\tilde{K} \triangleq HKH$, where $H = I - \mathbf{1}\mathbf{1}^\mathsf{T}/n$.

3. Compute a spectral decomposition $\tilde{K} = U\Lambda U^{\mathsf{T}}$.

4. Compute $U\Lambda^{1/2}$, whose columns will be the PC scores.

Figure 72 (see code below) shows an example of a dataset with two groups, with the (ordinary) PCA, and two KPCAs with polynomial kernel ($d = 4$), and Gaussian kernel ($\sigma = 1$). We notice that the polynomial kernel PCA does not separate the group linearly, but the Gaussian PCA does separate the group linearly. The choice of the kernel depends on the applications, and it is common to try out different kernels (and kernel parameters) to see which gives the "best" result.

```
x1 = matrix(rnorm(2*100), ncol=2)
theta <- runif(100, min=0, max=2*pi)
x2 <- ((3 + runif(100))*c(cos(theta), sin(theta))) %>%
    matrix(ncol=2, byrow=F)
X = rbind(x1, x2)
pch <- rep(c(1, 20), each=100)
plot(X, pch=pch, xaxt='n', yaxt='n', xlab='', ylab='')
prcomp(X)$x[,1:2] %>% plot(pch=pch, xaxt='n', yaxt='n',
                           xlab='PC1', ylab='PC2') ## looks the same as the data

## Polynomial kernel
quadratic <- function(x,y){
    (1+ sum(x*y))^4
}

## Gaussian kernel
gaussian <- function(x,y){
    exp(-sum((x-y)^2)/2)
}

kernel.pca <- function(X, kernel){
    K <- matrix(0, nrow=nrow(X), ncol=nrow(X))
    for(i in 1:nrow(K)){
        #print(sprintf("%d of %d", i, nrow(K)))
        for(j in i:ncol(K)){
            K[i,j] <- kernel(X[i,], X[j,])
        }
    }
    K <- K + t(K)
    diag(K) <- diag(K)/2
    #m <- nrow(K); kc <- t(t(K - colSums(K)/m) - rowSums(K)/m) + sum(K)/m^2
    K <- scale(K, scale=F)
    K <- scale(t(K), scale=F) %>% t

    K.eigen <- eigen(K, symmetric=TRUE)
    npos <- sum(K.eigen$values > 0)
```

```
    return(list(scores=K.eigen$vector[,1:npos]
           %*% diag(sqrt(K.eigen$value[1:npos])),
          variances=K.eigen$value[1:npos]))
}

X.kpca <- kernel.pca(X, quadratic)
X.kpca$scores[,1:2] %>% plot(pch=pch, xaxt='n', yaxt='n', xlab='KPC1', ylab='KPC2')

X.kpca <- kernel.pca(X, gaussian)
X.kpca$scores[,1:2] %>% plot(pch=pch, xaxt='n', yaxt='n', xlab='KPC1', ylab='KPC2')
```

## 8.2 Kernel Mean Embeddings

When working with normal data, the means and covariance matrices of data play a crucial role as they completely characterise the underlying distributions. For example, in Section 5.3.3, where we wanted to test whether two samples came from the same distribution, we tested whether or not they had the same mean. With more complex data types this can be insufficient, as it can happen that two different distributions have the same mean, even after assuming that they have the same covariance matrix.

Kernel mean embeddings work by mapping distributions to a more complex space, where the mean characterises the distribution. Once this is done, we are free to base our statistical analysis on the analysis of means, which is something that we are familiar with. Recalling our feature map $\Phi : \mathbb{R}^p \to \mathcal{H}$, we represent a random variable $\mathbf{X}$ by its kernel mean embedding

$$\mu_{\mathbf{X}} := \mathbb{E}\{\Phi(\mathbf{X})\},$$

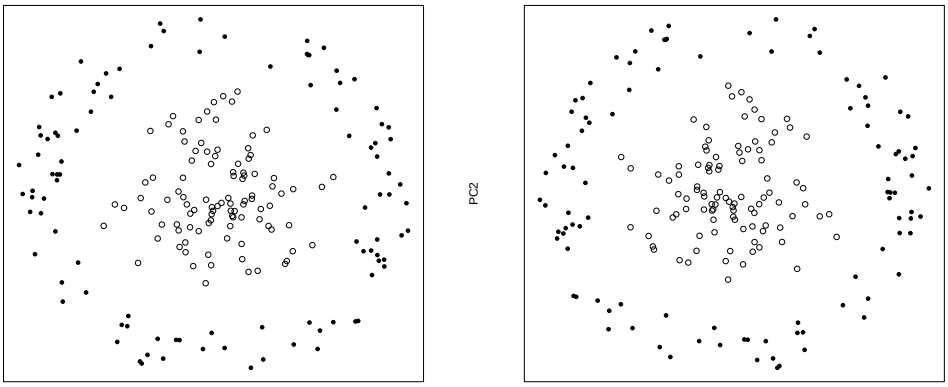
which is an element of $\mathcal{H}$. In fact, for some choices of $\mathcal{H}$, we may take $\Phi$ to be an injective mapping between distributions of $\mathbf{X}$ and elements of $\mathcal{H}$. As usual, we estimate this by the sample mean

$$\hat{\mu}_{\mathbf{X}} := \frac{1}{n}\sum_{i=1}^{n}\Phi(\mathbf{x}_i).$$

Now suppose that we have two samples $\mathbf{x}_1, \ldots, \mathbf{x}_m$ and $\mathbf{y}_1, \ldots, \mathbf{y}_n$, and we would like to know whether these two samples could have come from the same distribution. As in Section 5.3.3, we compare $\hat{\mu}_{\mathbf{X}}$ to $\hat{\mu}_{\mathbf{Y}}$ by looking at the size of $\hat{\mu}_{\mathbf{X}} - \hat{\mu}_{\mathbf{Y}}$. Here, this is represented by its norm in the space $\mathcal{H}$:

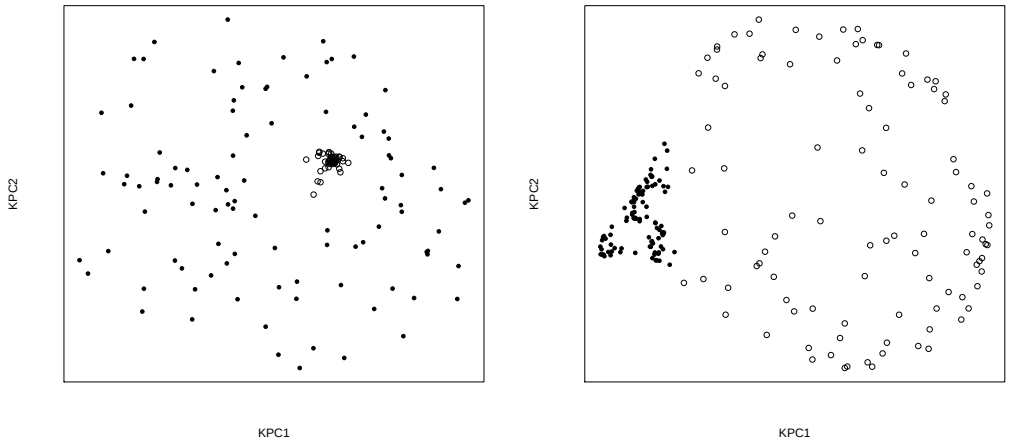
$$\|\hat{\mu}_{\mathbf{X}} - \hat{\mu}_{\mathbf{Y}}\|_{\mathcal{H}}^2 = \langle \hat{\mu}_{\mathbf{X}} - \hat{\mu}_{\mathbf{Y}}, \hat{\mu}_{\mathbf{X}} - \hat{\mu}_{\mathbf{Y}} \rangle_{\mathcal{H}}.$$

As with KPCA, we can derive a simple expression for this which only depends on $\Phi$ through the kernel function $k$.




(a) Original data (from 2 groups).

(b) (ordinary) PCA of the data.




(c) Kernel PCA of the data using a polynomial kernel.

(d) Kernel PCA of the data using a Gaussian kernel.

Figure 72: Illustration of the advantage of kernel PCA for data with multiple groups that are not linearly separable. Notice that the Gaussian kernel PCA separates linearly the two groups.

**Proposition 8.2.1.** We have that

$$\|\hat{\mu}_{\mathbf{X}} - \hat{\mu}_{\mathbf{Y}}\|^2_{\mathcal{H}} = \frac{1}{m^2}\sum_{i,j=1}^{m} k(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{y}_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(\mathbf{y}_i, \mathbf{y}_j).$$

*Proof.* See video. □

This measure of the difference between two distributions can be used to derive hypothesis tests, but the theory of this is beyond the scope of the course. For a biological data example in R, see

`https://bioconductor.org/packages/release/bioc/html/MMDiff2.html`,

for an image analysis example in Python, see

`https://docs.seldon.io/projects/alibi-detect/en/latest/examples/cd_mmd_cifar10.html`,

and for a text analysis example in Python, see

`https://docs.seldon.io/projects/alibi-detect/en/stable/examples/cd_text_imdb.html`.

A simple implementation in R is given by `kmdd` in the package `kernlab`.

# 9 Advanced Topic (only ST412): Multidimensional Scaling

Suppose that you are given a matrix containing distances (or, alternatively, similarities) between individuals. *Multidimensional scaling* (MDS) concerns representing the individuals in a low-dimensional space, so that Euclidean distances in the low-dimensional space are as similar as possible to the original distances. For instance, given the distances between European capitals (in miles), we would hope to reconstruct a map of Europe.

There are many practical applications where one might have to deal with distances or similarities rather than actual data for each individual. One possibility is that only distances are available (or easily obtained). Another possibility is that, even when data for individuals are available, we wish to define our own measure of distance to reflect certain aspects of the problem we're dealing with. Going back to the previous example, we may be more interested in travelling time between cities rather than physical distances. In this case we would define a distance matrix based on time, which might give rise to a different map from when using miles.

Beyond these simple examples, in many real-life applications it can be quite important how we define distances. Multidimensional scaling offers great flexibility, as it doesn't force us to use any pre-specified distance metric.

## 9.1 Introduction

The goal of classical multidimensional scaling is, starting from a dissimilarity matrix $\Delta = (\delta_{ij})$, where $\delta_{ij}$ is the dissimilarity between the $i$th and $j$th individual or observation (you think of a dissimilarity as a distance, but without the constraints of a metric distance), to represent points in a low dimensional space $\mathbb{R}^k$ such that the Euclidean distances between points approximate $\Delta$. We assume that the number of dimensions $k$ is given. Let $D = (d_{ij})$ be these Euclidean distances. A natural objective would be to look at how close $D$ and $\Delta$ are. One obvious way of measuring this is to compute the sum of squared differences between the dissimilarities $\Delta$ and the Euclidean distances $D$, in which case the goal becomes to minimise the so-called *raw stress function* ('stress' stands for 'standardized residual sum of squares'):

$$\min_D \sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij} - d_{ij})^2 = \|\Delta - D\|^2. \tag{9.1.1}$$

Because the value of the raw stress depends on the units of measurement, it is common to report the *stress-1 function*, which is simply the following standardized version of the raw stress

$$\min_D \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij} - d_{ij})^2}{\sum_{i=1}^{n}\sum_{j=1}^{n}\delta_{ij}^2}. \tag{9.1.2}$$

The denominator in (9.1.2) does not depend on $d_{ij}$, so we may equivalently minimize (9.1.1). The appealing property of (9.1.2) is that its values lie in $[0, 1]$, so it can be easily interpreted.

Without any constraints on $D$, both (9.1.1) and (9.1.2) would be solved by $D = \Delta$. If there are only rank constraints on $D$, then the Young–Eckart–Mirsky Theorem (Theorem 2.2.3) tells us that $D$ is given by a truncation of the singular value decomposition of $\Delta$. That is, if $\Delta = \sum_{i=1}^{n} l_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}$, then the solution of (9.1.1) over all matrices $D$ of rank at most $k$ is $D = \sum_{i=1}^{k} l_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}$. Unfortunately, this solution does not necessarily reflect the pairwise distances of $n$ points in $\mathbb{R}^q$, for some $q \geq 1$, and therefore (9.1.1) cannot be solved directly using these results, but needs to be solved using numerical methods, such as gradient descent. Nevertheless, there is a way of getting a closed form solution to MDS if we take the objective function $\|H\Delta_2 H - HD_2H\|_F$, where $\Delta_2 = (\delta_{ij}^2)$, $D_2 = (d_{ij}^2)$, $\|\cdot\|_F$ is the Frobenius norm (see page 17) and $H$ is the $n \times n$ centering matrix. This is called *classical scaling*, or *classical multidimensional scaling*.

## 9.2 Classical scaling (also called classical MDS)

Let $\mathtt{X}$ be an $n \times k$ matrix, which we will assume to be column-centered (without loss of generality, as Euclidean distances are invariant to translations). The squared Euclidean distance between rows $\mathbf{x}_i$ and $\mathbf{x}_j$ is $d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}(\mathbf{x}_i - \mathbf{x}_j) = \mathbf{x}_i^\mathsf{T}\mathbf{x}_i + \mathbf{x}_j^\mathsf{T}\mathbf{x}_j - 2\mathbf{x}_i^\mathsf{T}\mathbf{x}_j$. The matrix of Euclidean distances can be written as

$$
\begin{aligned}
D_2 &= \mathbf{1}(\mathbf{x}_1^\mathsf{T}\mathbf{x}_1, \ldots, \mathbf{x}_n^\mathsf{T}\mathbf{x}_n) + \begin{pmatrix} \mathbf{x}_1^\mathsf{T}\mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n^\mathsf{T}\mathbf{x}_n \end{pmatrix} \mathbf{1}^\mathsf{T} - 2 \begin{pmatrix} \mathbf{x}_1^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_1^\mathsf{T}\mathbf{x}_2 & \ldots & \mathbf{x}_1^\mathsf{T}\mathbf{x}_n \\ \ldots & & & \\ \mathbf{x}_n^\mathsf{T}\mathbf{x}_1 & \mathbf{x}_n^\mathsf{T}\mathbf{x}_2 & \ldots & \mathbf{x}_n^\mathsf{T}\mathbf{x}_n \end{pmatrix} \\
&= \mathbf{1}\mathbf{s}^\mathsf{T} + \mathbf{s}\mathbf{1}^\mathsf{T} - 2\mathtt{X}\mathtt{X}^\mathsf{T} && (9.2.1) \\
&= \mathbf{1}\mathbf{s}^\mathsf{T} + \mathbf{s}\mathbf{1}^\mathsf{T} - 2S, && (9.2.2)
\end{aligned}
$$

where $\mathbf{1}$ is an $n \times 1$ vector with ones, $\mathbf{s}$ is a vector of squared lengths, and $S = \mathtt{X}\mathtt{X}^\mathsf{T}$ is an inner-product matrix measuring similarities (with $\mathrm{diag}(S) = \mathbf{s}$). We note that, because $\mathtt{X}$ is column-centered, $S$ is also column-centered (and row-centered, as it's symmetric). To see this, the sum of the $j^{th}$ column in $S$ is $\sum_{i=1}^{n} \mathbf{x}_i^\mathsf{T}\mathbf{x}_j = \left(\sum_{i=1}^{n} \mathbf{x}_i^\mathsf{T}\right)\mathbf{x}_j = \mathbf{0}^\mathsf{T}\mathbf{x}_j = 0$.

Recall the *centering matrix* $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}$ (we have seen that $HA$ is equal to zero-centering the columns in $A$). Then from (9.2.2),

$$
HD_2H = H\mathbf{1}\mathbf{s}^\mathsf{T}H + H\mathbf{s}\mathbf{1}^\mathsf{T}H - 2HSH = \mathbf{0}\mathbf{s}^\mathsf{T}H + H\mathbf{s}0 - 2S = -2S,
$$

since centering a vector of 1's gives 0's and $S$ was already centered so that $HSH = S$. In words, a doubly-centered distance matrix $D_2^* = HD_2H$ is equal to minus twice the inner-products $-2S = -2\mathtt{X}\mathtt{X}^\mathsf{T}$. We can therefore try to find $\mathtt{X}$ directly from $-\frac{1}{2}HD_2H$. Since $D_2$ is not known, we can replace it by $\Delta_2$, and therefore get the following algorithm for computing $\mathtt{X}$.

**Algorithm 9.2.1** (Classical MDS)**.** Let $\Delta = (\delta_{ij})$ be an $n \times n$ dissimilarity matrix, which we wish to approximate using a $k$-dimensional space. Let $\Delta_2 = (\delta_{ij}^2)$.

1. Find the associated similarity matrix $S_{\Delta_2} = -\frac{1}{2} H \Delta_2 H$
2. Find the eigendecomposition $S_{\Delta_2} = E \Lambda E^\mathsf{T}$
3. Select the $k$ largest eigenvalues. Let $\Lambda_k$ be the corresponding submatrix of $\Lambda$ and $E_k$ the corresponding columns in $E$. Then the coordinates are given by $\mathtt{X}^{(k)} = E_k \left(\Lambda_k\right)^{1/2}$.

Notice that the Eckart–Young–Mirsky Theorem (Theorem 2.2.3) implies that this algorithm solves the problem of finding the $n \times k$ matrix $\mathtt{X}$ such that

$$\|S_{\Delta_2} - \mathtt{X}\mathtt{X}^\mathsf{T}\|_F$$

is minimized.

**Example 9.2.1.** We consider the dataset `eurodist` which measures the road distances (km) between 21 European cities. We apply classical scaling to the distance matrix in order to produce a 2-dimensional map. The result (after flipping the $y$-axis, which does not change the solution in terms of distances) is shown in Figure 73. The figure highly resembles the map of Europe. When producing the plot, it is important to set the aspect ratio of the x and y-axes to 1, to that we can correctly perceive Euclidean distances by looking at the map.

The stress-1 function is equal to 0.0081, a very small value that indicates the approximation is of very high quality. This appears reasonable, as we are working with geographical distances.

```
loc <- cmdscale(eurodist,k=2)
x <- loc[, 1]
y <- -loc[, 2] #flip so North is at the top

## asp=1 ensures Euclidean dist represented correctly
plot(x,y,type="n",xlab="",ylab="",asp=1,axes=FALSE,
  main="Classical")
text(x,y,rownames(loc),cex=0.8)

## compute stress-1 at the optimum
dexact <- as.matrix(eurodist)
dexact <- as.vector(dexact[upper.tri(dexact)])
dapprox <- dist(loc,method='euclidean')
dapprox <- as.matrix(dapprox)
dapprox <- as.vector(dapprox[upper.tri(dapprox)])
sum((dexact-dapprox)^2)/sum(dexact^2)

## [1] 0.008125444
```

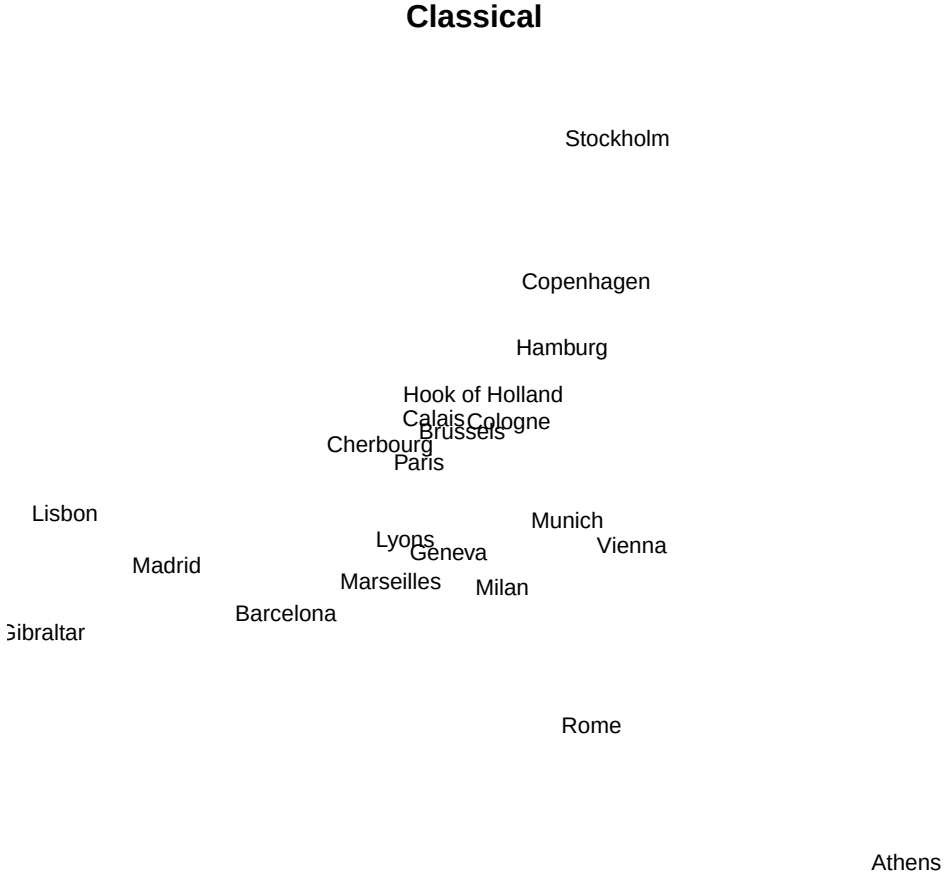



Figure 73: Maps approximating road distances between European cities using classical MDS.

**Proposition 9.2.2 (Classical MDS).** Let $\mathtt{Y}$ be an $n \times p$ matrix, and suppose that $\Delta$ was originally obtained by computing Euclidean distances between rows in $\mathtt{Y}$,

$$\Delta_2 = \mathbf{1}\mathbf{s}_y^\mathsf{T} + \mathbf{s}_y\mathbf{1}^\mathsf{T} - 2\mathtt{Y}\mathtt{Y}^\mathsf{T}$$

Then the $k$-dimensional classical scaling solution is equivalent to plotting the first $k$ principal components scores of $\mathtt{Y}$.

The proof follows immediately from Algorithm 9.2.1 and derivations leading to it, after noting that $S_{\Delta_2} = -\frac{1}{2}H\Delta_2 H = \mathtt{Y}\mathtt{Y}^\mathsf{T}$, The implication of Proposition 9.2.2 is that classical scaling can be seen as a generalization of PCA. PCA implicitly defines Euclidean distances between rows, while MDS can use any other user-defined distance. It also gives a nice interpretation of the principal components plot as a map where distances approximate the Euclidean distances computed on the full data.

## 9.3 Beyond classical scaling

There are several popular extensions of classical scaling which, instead of minimizing (9.1.1), target other functions. While these alternatives offer greater flexibility, in general there is not a closed-form solution and one must resort to numerical optimization methods. Here we shall not worry about how one would actually implement the optimization; just assume that there is adequate software for this purpose.

One family of methods generalizing classical scaling is called *metric scaling*. As in classical scaling, the stress function tries to approximate the exact value of $d_{ij}$. Two such scaling methods focus on the *stress-2 function* or on relative errors (the latter being also called *Sammon scaling*). The *stress-2* function is the $R^2$ coefficient (*coefficient of determination*) between the original and approximated distances. Of course, here the goal is to maximize (rather than minimize) *stress-2*.

**Definition 9.3.1 (Stress-2 function).**

$$\max \frac{\left(\sum_{i<j}(d_{ij} - \bar{d})(\delta_{ij} - \bar{\delta})\right)^2}{s_d^2 s_\delta^2}, \tag{9.3.1}$$

where $\bar{d}, \bar{\delta}$ are the means across $d_{ij}$ and $\delta_{ij}$ (respectively), and $s_d^2, s_\delta^2$ the variances.

The stress-2 function can be interpreted as the proportion of variability in $d_{ij}$ explained by $\delta_{ij}$. Similar to stress-1, it is guaranteed to be in $[0, 1]$. This is not the case for the target function in *Sammon mapping*, which we define below.

**Definition 9.3.2 (Sammon and elastic scaling).** The goal in Sammon scaling is to minimize

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}}. \tag{9.3.2}$$

Elastic scaling minimizes relative errors

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}^2}. \tag{9.3.3}$$

We comment on differences between (9.1.1), (9.3.1) and (9.3.2). Both stress-1 and stress-2 minimize absolute errors, which typically makes the solutions sensitive to large values of $d_{ij}$ and $\delta_{ij}$. By giving larger weight to individuals who are far away from each other, the map given by (9.3.1) tends to approximate large $d_{ij}$ more accurately than small $d_{ij}$. In particular, if there are outliers in $d_{ij}$ these can have a substantial effect on the solution. In contrast, (9.3.2)-(9.3.3) down-weight large $\delta_{ij}$ (in particular, (9.3.3) minimizes squared relative errors $(1 - d_{ij}/\delta_{ij})^2$). As a consequence, maps produced by (9.3.2)-(9.3.3) tend to represent small $d_{ij}$ more accurately than large $d_{ij}$, and to be less sensitive to outliers.

All methods we have seen so far fall within the family of metric scaling. *Non-metric scaling methods* is an alternative family of methods which, rather than trying to approximate the exact distances $d_{ij}$, approximate a monotone increasing transformation $h(\delta_{ij})$ of the dissimilarities.

**Definition 9.3.3 (Non-metric scaling).** The goal in non-metric scaling is to minimize

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (h(\delta_{ij}) - d_{ij})^2 \tag{9.3.4}$$

with respect to both $d_{ij}$ and $h(\cdot)$, where the latter is constrained to be an increasing function.

That is, *non-metric scaling* reflects which objects are closer and which are farther from each other, without so much emphasis on the exact values of the distances. A common approach to non-metric scaling is to sequentially apply the following two steps, until the solution is deemed to have converged.

**Algorithm 9.3.1** (Non-metric scaling)**.** Initialize $\hat{h}(\delta_{ij}) = \delta_{ij}$, the identity function.

1. Update $\tilde{\delta}_{ij} = \hat{h}(\delta_{ij})$. Find $d_{ij}$ minimizing

$$\sum_{i,j}(\tilde{\delta}_{ij} - d_{ij})^2$$

which is a classical scaling problem with modified distances $\tilde{\delta}_{ij}$.

2. Update $\hat{h}(\cdot)$ by fitting a regression model $d_{ij} = h(\tilde{\delta}_{ij}) + e_{ij}$, where $(\tilde{\delta}_{ij}, d_{ij})$ are as obtained in Step (1), $h(\cdot)$ is restricted to be increasing and $e_{ij}$ are the residuals. This can be done for instance with a technique called *monotone regression*.

Repeat Steps 1. and 2. until convergence, *i.e.* the $d_{ij}$ change little between two consecutive iterations.

**Example 9.3.4.** We reconsider Example 9.2.1 with road distances between European cities. We produce new maps with Sammon scaling and isoMDS (a type of non-metric scaling), shown in the top and bottom panels of Figure 74. All three solutions are quite similar, but some differences can be observed. For instance, the distance between Munich and Vienna is larger in the middle panel than in the other two panels.

```
library(MASS)
loc2 <- sammon(eurodist, k=2)

## Initial stress        : 0.01705
## stress after  10 iters: 0.00951, magic = 0.500
## stress after  20 iters: 0.00941, magic = 0.500

x <- loc2$points[,1]
y <- -loc2$points[,2]
op <- par(mfrow=c(2,1))
plot(x,y, type="n", xlab="", ylab="", asp=1, axes=FALSE, main="Sammon")
text(x,y, labels=rownames(loc), cex=0.8)

#isoMDS
loc4 <- isoMDS(eurodist,k=2)

## initial  value 7.505733
## final  value 7.505688
## converged

x <- loc4$points[,1]; y <- -loc4$points[,2]
plot(x,y, type="n", xlab="", ylab="", asp=1, axes=FALSE, main="isoMDS")
text(x,y, labels=rownames(loc), cex=0.8)
par(op)
```

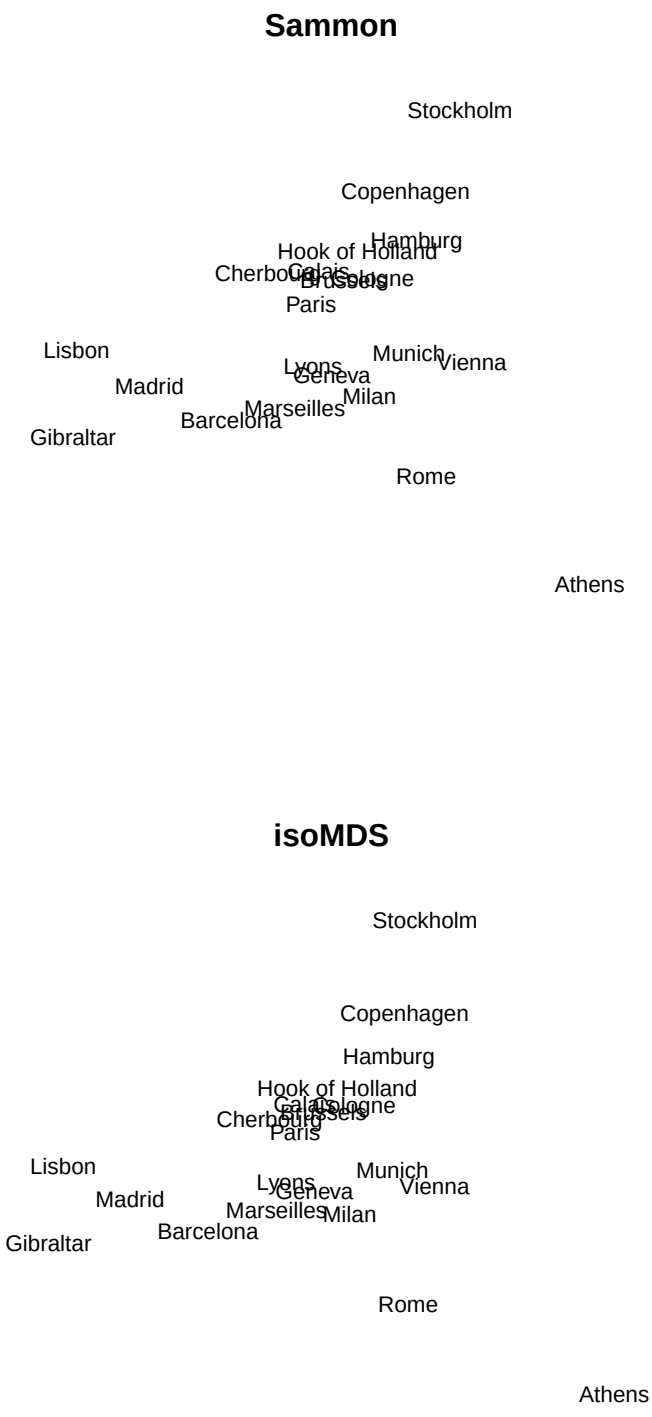



Figure 74: Maps approximating road distances between European cities. Top: Sammon scaling; Bottom: non-metric isoMDS

## 9.4 Quality of the approximation and number of dimensions

So far we discussed how to produce a $k$-dimensional representation of a distance matrix. Obviously, the larger $k$ the greater the quality of the approximation, but the harder it becomes to convey the results. In practice, choosing $k$ is a trade-off between representation accuracy and practical considerations.

In order to measure the quality of the approximation we may check the value of the target function, perhaps plotted versus $k$. Another popular option is to produce *Shepard plots*, which plot $d_{ij}$ against $\delta_{ij}$. Because of the connection between PCA and classical scaling, when the input distances $\delta_{ij}$ are Euclidean we may also examine the eigenvalues in a *scree plot* and compute the proportion of explained variability as we would normally do for PCA.

**Example 9.4.1.** We continue Examples 9.2.1 and 9.3.4 with road distances between European distances.

Figure 75 shows the Shepard plots. As expected, classical scaling approximates better large distances than Sammon scaling, but the approximation for small distances is worse. Non-metric MDS behaves similar to classical scaling in this example.

```
dapprox2 <- dist(loc2$points,method='euclidean')
dapprox2 <- as.matrix(dapprox2)
dapprox2 <- as.vector(dapprox2[upper.tri(dapprox2)])

dapprox4 <- dist(loc4$points,method='euclidean')
dapprox4 <- as.matrix(dapprox4)
dapprox4 <- as.vector(dapprox4[upper.tri(dapprox4)])

op <- par(mfrow=c(3,1))
plot(dapprox, dexact, main='Classical')
abline(0,1)
plot(dapprox2, dexact, main='Sammon')
abline(0,1)
plot(dapprox4, dexact, main='isoMDS')
abline(0,1)
par(op)
```

We now examine what might be an adequate value of $k$. For $k = 1$ the classical scaling stress function is 0.131, which decreases substantially to 0.0081 for $k = 2$ and then stays almost constant at 0.0079 for $k = 3$. This would suggest to choose $k = 2$, which is in agreement with our intuition that kilometric distances on a relatively small region are well approximated in two-dimensions. For instance, if we considered cities around all the World we might need 3 dimensions to reflect that the Earth is not flat!

```
kseq <- 1:3
stress <- rep(NA,length(kseq))
for (i in 1:length(kseq)) {
```

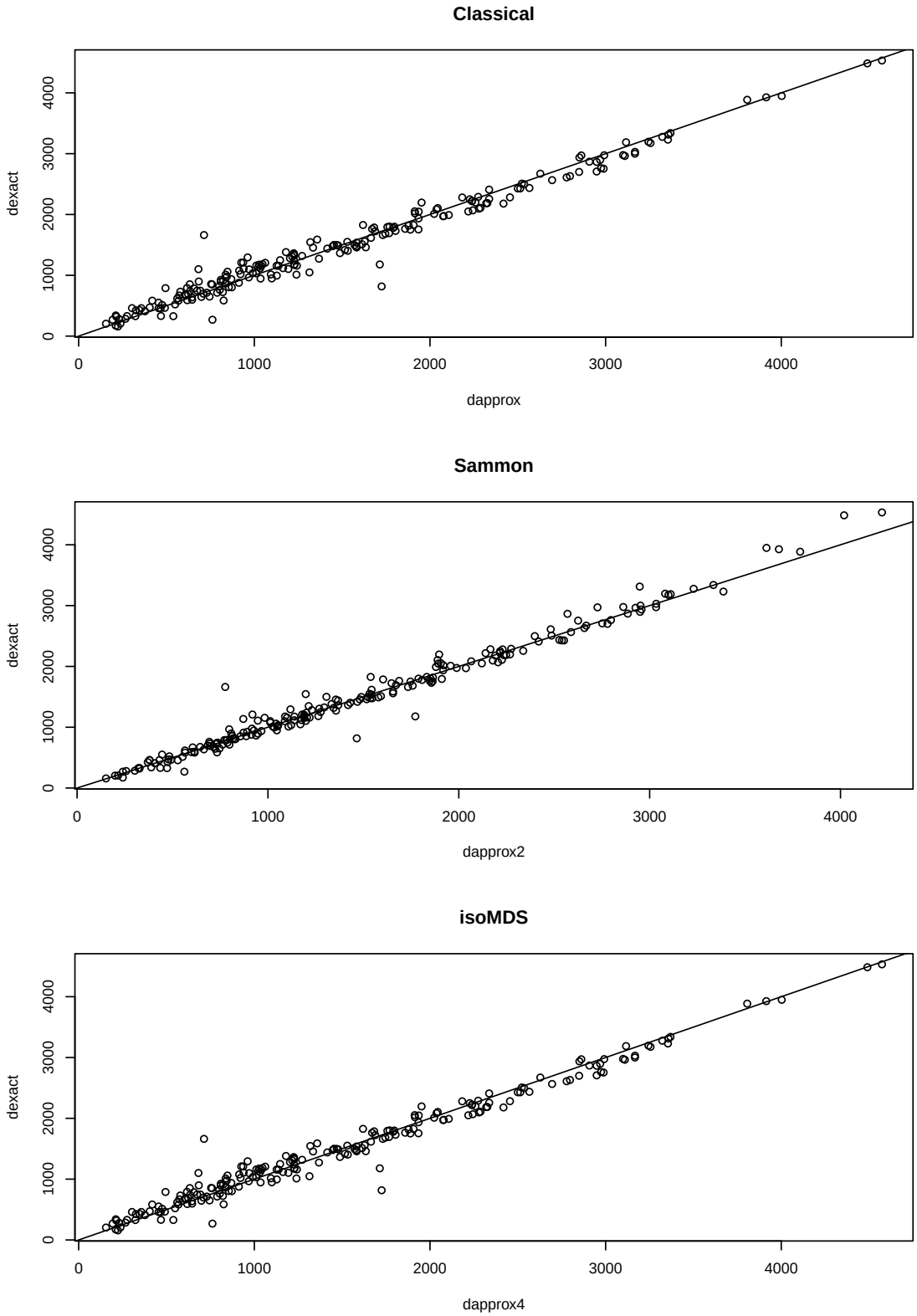



Figure 75: Shepard plots for classical (top), Sammon (middle) and non-metric (bottom) scaling.

```
  loc <- cmdscale(eurodist,k=kseq[i])
  dapprox <- dist(loc,method='euclidean')
  dapprox <- as.matrix(dapprox)
  dapprox <- as.vector(dapprox[upper.tri(dapprox)])
  stress[i] <- sum((dexact-dapprox)^2)/sum(dexact^2)
}
stress

## [1] 0.131539705 0.008125444 0.007955413
```

---