Multivariate statistics

27 maggio 2025

Indice

Capitolo 1

Introduction

Remark 1 (Prerequisites). Random variables/vectors, linear models (least square principle, assumptions, R^2), matrix algebra (matrices, determinant, inverse, quadratic forms, eigenvalues-eigenvectors, Lagrange multipliers)

Remark 2 (Multivariate analysis situation). Large number of variables (p) observed on the same set of statistical units (n). Aim is to summarize somehow the data.

Important remark 1. Methods tackled in the course:

- *dimension reduction methods*: they share the aim to summarize the information contained in our data building new variables, limited in number, which can convey as much info as the original set. Among these the two methods we see are:
  - *Principal component*: is data transformation method. We transform data to obtain meaningful summary. Always feasible.
  - *Factor analysis*: is a *model* where we have to make assumptions: we fit and then check if model adequately is met.
- *discriminant analysis*: a supervised classification method. We have two (or more than two) groups and know that groups are different by characteristics (eg. healthy/diseased) collected in both groups. We want to find
  - a way to characterize the two groups: how different are which respect to which characteristic
  - a criteria to separate the two groups; to find a rule to assign a new observation (whose group membership is unknown) to one or the other group.
- *cluster analysis*: our aim is to find/create groups in the data (we have no previous groups as in discriminant analysis). Splitting should be done in order to have homogeneous units within group and different between groups

Important remark 2. Notation: for quantities referring to the population we use greek letters, while roman letters will be used for sample quantities. At the population level:

- the equivalent of S (variance/covariance matrix) is denoted as $\mathbf{\Sigma}$, which is called **population covariance matrix**
- the equivalent of the mean vector $\overline{x}$ will be μ
- the population analog of $\mathbf{R}$ (correlation matrix) is denoted by $\boldsymbol{\rho}$, the population correlation matrix

1.1 Matrices

1.1.1 Data matrix

The data matrix includes all data on observation were interested in:

- denoted as $\mathbf{X}$, it is $n \times p$ with n units (the generic one is unit j) and p variables (the generic one being i), be they categorical or numeric;
- each row correspond to a different unit;
- each column correspond to a variable: $X_1, \ldots, X_p$;

$$\underset{n\times p}{\mathbf{X}} = \begin{bmatrix} x_{11} & \ldots & x_{1i} & \ldots x_{1p} \\ \ldots & \ldots & \ldots & \ldots \\ x_{j1} & \ldots & x_{ji} & \ldots x_{jp} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & \ldots & x_{ni} & \ldots x_{np} \end{bmatrix}$$

Remark 3. Multivariate analysis is concerned either with:

- studying the *relationships between variables* (PCA, Factor analysis and LDA focus on relationships between *columns*)
- studying the *similarities between units* (cluster analysis focus on the *row* of the data matrix).

Remark 4. Starting from the data matrix $\mathbf{X}$ a series of different matrices can be derived. We will first concentrate on the matrices dealing with relationships between variables, leaving the theme of measuring dissimilarities between units to when we will deal with clustering.

1.1.2 Mean vector

Important remark 3. For the moment we assume we're dealing with *numeric variables*.

We can associate to the data matrix the vector of means, $\overline{\mathbf{x}}$, a $p \times 1$ column vector containing column means:

$$\underset{p\times 1}{\overline{\mathbf{x}}} = \begin{bmatrix} \overline{x}_1 \\ \vdots \\ \overline{x}_i \\ \vdots \\ \overline{x}_p \end{bmatrix} = \left(\frac{\mathbf{1}_n^\top \mathbf{X}}{n}\right)^\top = \left(\frac{1}{n}\mathbf{1}_n^\top \mathbf{X}\right)^\top = \frac{1}{n}\mathbf{X}^\top \mathbf{1}_n$$

To obtain it in matrix form by pre-multiplying $\mathbf{X}$ by a row vector of all 1 we obtain the sum of all elements in column, then we divide by n to have the mean and finally we transpose to have a column vector.

1.1.3 Mean centered matrix

In certain applications it might be useful to express the variables as deviations from the mean. The mean vector can be used to obtain the mean centered data matrix, $\tilde{\mathbf{X}}$:

$$\underset{n\times p}{\tilde{\mathbf{X}}} = \begin{bmatrix} \tilde{x}_{11} & \dots & \tilde{x}_{1i} & \dots & \tilde{x}_{1p} \\ \dots & \dots & \dots & \dots & \\ \tilde{x}_{j1} & \dots & \tilde{x}_{ji} & \dots & \tilde{x}_{jp} \\ \dots & \dots & \dots & \dots & \\ \tilde{x}_{n1} & \dots & \tilde{x}_{ni} & \dots & \tilde{x}_{np} \end{bmatrix} = \begin{bmatrix} x_{11}-\overline{x}_1 & \dots & x_{1i}-\overline{x}_i & \dots & x_{1p}-\overline{x}_p \\ \dots & \dots & \dots & \dots & \dots \\ x_{j1}-\overline{x}_1 & \dots & x_{ji}-\overline{x}_i & \dots & x_{jp}-\overline{x}_p \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1}-\overline{x}_1 & \dots & x_{ni}-\overline{x}_i & \dots & x_{np}-\overline{x}_p \end{bmatrix}$$

How to obtain $\tilde{\mathbf{X}}$ from $\mathbf{X}$ and $\overline{\mathbf{x}}$? The difference between two matrix can be obtained only if we have matrix of the same size; we need to subtract from $\mathbf{X}$ a matrix $\overline{\mathbf{X}}$ where each row is a copy of $\overline{\mathbf{x}}$:

$$\underset{n\times p}{\overline{\mathbf{X}}} = \underset{n\times 1}{\mathbf{1_n}}\underset{1\times p}{\overline{\mathbf{x}}^\top} = \begin{bmatrix} \overline{x}_1 & \dots & \overline{x}_i & \dots & \overline{x}_p \\ \dots & \dots & \dots & \dots & \dots \\ \overline{x}_1 & \dots & \overline{x}_i & \dots & \overline{x}_p \\ \dots & \dots & \dots & \dots & \dots \\ \overline{x}_1 & \dots & \overline{x}_i & \dots & \overline{x}_p \end{bmatrix}$$

Therefore

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1_n}\overline{\mathbf{x}}^\top = \mathbf{X} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\mathbf{X} = \underbrace{\left(\mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)}_{\mathbf{A}}\mathbf{X}$$

where in the final passage we collected $\mathbf{X}$ (to the right) and added the identity matrix which leaves the product unchanged.

Remark 5. We have that:

- Mean vector of $\tilde{\mathbf{X}}$ is $\mathbf{0}$
- $\tilde{\mathbf{X}}$ defines a translation of the origin of the original reference system. The shape of the point cloud remains unchanged, but the origin of the axes is moved to $\overline{\mathbf{x}}$.

Definition 1.1.1 (Centering matrix)**.** $\mathbf{A}$ is usually called the *centering matrix*: if we pre-multiply a matrix for $\mathbf{A}$ we obtained the mean centered matrix;

Important remark 4. $\mathbf{A}$ is very interesting from the algebraic point of view:

- is $n \times n$;
- is *symmetric*: coincides with its transpose since difference of two symmetric matrix ($\mathbf{1}_n\mathbf{1}_n^\top$ transposed is the same and $\mathbf{I}_n$ is symmetric)
- is *idempotent*: $\mathbf{A} = \mathbf{A}^2$

1.1.4 Standardized data matrix

If one wants to eliminate the effect of different scales on the observed variables, one can resort on this matrix.
To standardize a single cell we have

$$z_{ji} = \frac{x_{ji} - \overline{x}_i}{s_i}$$

where the numerator is clearly taken from $\tilde{X}$ while at the denominator we have the standard deviation of i-th column/variable. To compute in matrix form how to divide each column by its standard deviation?
First we build a diagonal matrix which have the variances s_i^2 on the main diagonal

$$\underset{p\times p}{\mathbf{D}} = \begin{bmatrix} s_1^2 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & s_i^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & s_p^2 \end{bmatrix}$$

then we define its power of $-1/2$

$$\underset{p\times p}{\mathbf{D}^{-\frac{1}{2}}} = \begin{bmatrix} \frac{1}{s_1} & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \frac{1}{s_i} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \frac{1}{s_p} \end{bmatrix}$$

Thus we obtain the standardized data matrix post-multiplying the standardized data matrix by $\mathbf{D}^{-\frac{1}{2}}$ (equivalent to dividing each column by its standard deviation)

$$\underset{n\times p}{\mathbf{Z}} = \underbrace{\begin{bmatrix} x_{11} - \overline{x}_1 & \dots & x_{1i} - \overline{x}_i & \dots & x_{1p} - \overline{x}_p \\ \dots & \dots & \dots & \dots & \dots \\ x_{j1} - \overline{x}_1 & \dots & x_{ji} - \overline{x}_i & \dots & x_{jp} - \overline{x}_p \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} - \overline{x}_1 & \dots & x_{ni} - \overline{x}_i & \dots & x_{np} - \overline{x}_p \end{bmatrix}}_{\tilde{\mathbf{X}}} \underbrace{\begin{bmatrix} \frac{1}{s_1} & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \frac{1}{s_i} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \frac{1}{s_p} \end{bmatrix}}_{\mathbf{D}^{-\frac{1}{2}}}$$

Thus, at the same time

$$\underset{n\times p}{\mathbf{Z}} = \tilde{\mathbf{X}}\mathbf{D}^{-\frac{1}{2}} = \mathbf{A}\mathbf{X}\mathbf{D}^{-\frac{1}{2}}$$

Remark 6. Each column of $\mathbf{Z}$ has zero mean and unit variance.

Remark 7. We can work with raw, mean centered or standardized variables: which matrix is better to use is something we'll discuss.

1.1.5 (Variance-)Covariance matrix

Remark 8. The covariance between two variables X_k and X_h is defined as

$$\mathrm{Cov}\,(X_k, X_h) = \frac{\sum_{j=1}^{n}(x_{jk} - \overline{x}_k)(x_{jh} - \overline{x}_h)}{n} = \frac{1}{n}\sum_{j=1}^{n}(\tilde{x}_{jk})(\tilde{x}_{jh}) = \frac{1}{n}\tilde{\mathbf{x}}_k^\top \tilde{\mathbf{x}}_h$$

where $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}_h$ are the k-th and h-th columns of $\tilde{\mathbf{X}}$ respectively.

Important remark 5. It is worth remembering that

- $\text{Cov}(X_k, X_h) = \text{Cov}(X_h, X_k)$
- the covariance between a variable and itself is but the variance of the variable itself
- numerator of variance and covariance (excluding $\frac{1}{n}$) are called *deviance* and *codeviance*

Variances and covariances can then be summarized in the so called covariance matrix $\mathbf{S}$, a $p \times p$ matrix that has variances on the main diagonal and covariances outside

$$\underset{p\times p}{\mathbf{S}} = \begin{bmatrix} \text{Var}[X_1] & \dots & \text{Cov}(X_1, X_j) & \dots & \text{Cov}(X_1, X_p) \\ & & \text{Var}[X_i] & \dots & \text{Cov}(X_i, X_p) \\ & & & & \text{Var}[X_p] \end{bmatrix} = \begin{bmatrix} s_1^2 & \dots & s_{1i} & \dots & s_{1p} \\ & & s_i^2 & \dots & s_{ip} \\ & & & & s_p^2 \end{bmatrix}$$

If we have p variables:

- the number of variances is obviously p (on the diagonal)
- the number of (unique) covariances, which fill the upper matrix, is the sum of the first p natural number which is

$$ncovs = \frac{p(p-1)}{2}$$

  eg with 3 variables we have 3 unique covariances

Important remark 6 (Matrix form derivation). Let's see the component for a generic element on the diagonal (variance) and out of it (covariance) respectively

$$\text{Var}[X_i] = s_i^2 = \frac{\sum_{j=1}^n (x_{ji} - \overline{x}_i)^2}{n}, \quad \text{Cov}(X_i, X_l) \quad = s_{il} = \frac{\sum_{j=1}^n (x_{jl} - \overline{x}_l)(x_{ji} - \overline{x}_i)}{n}$$

To obtain $\mathbf{S}$ the main ingredient are the element of matrix $\tilde{\mathbf{X}}$ of deviations from the mean, which are multiplied by themselves and then divided by n so

$$\mathbf{S} = \frac{1}{n}\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$$

Remark 9. Generally $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ is called **deviance/codeviance matrix** (numerators of variances/covariances)

Remark 10. Here we used the biased version of the matrix (with n at denominator of each element, not $(n-1)$. We could define as well

$$\widehat{\mathbf{S}} = \frac{1}{n-1}\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$$

which is the *unbiased covariance matrix*.
For our purposes using one or the other is the same but if needed to change between the two is done easily by multiplying by n and divide by $n-1$ or viceversa

Important remark 7 (Properties). It's

- *squared* $p \times p$
- *symmetric* (so people usually write just the upper triangular part): we know this from property of covariance but as well if we transpose above we obtain the same
- is *positive semi definite*
- it's trace is the called t*otal variance*:

$$\operatorname{Tr} \mathbf{S} = \sum_{i=1}^{p} \operatorname{Var}[X_i]$$

Dimostrazione. To prove a matrix is positive semi definite we can procede in two ways

- prove its eigenvalues are $>= 0$
- prove the determinant of the matrix and determinant of all the minors are $>= 0$

To prove $\mathbf{S}$ is positive semi-definite go this second way. In the simple case of just two variables, let's consider just two mean-centered variables

$$\mathbf{S} = \frac{1}{n}\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} = \frac{1}{n}\begin{bmatrix} Dev(X_1) & Codev(X_1, X_2) \\ Codev(X_1, X_2) & Dev(X_2) \end{bmatrix}$$

where $Dev(X_1)$ is the deviance (numerator of the variance) while $Codev(X_1, X_2)$ is the codeviance (numerator of covariance).
For what concerns:

- determinant of $\mathbf{S}$:

$$\det \mathbf{S} = Dev(X_1) \cdot Dev(X_2) - Codev^2(X_1, X_2)$$

To check the inequality

$$\begin{aligned} &Dev(X_1) \cdot Dev(X_2) - Codev^2(X_1, X_2) \geq 0 \\ &Dev(X_1) \cdot Dev(X_2) \geq Codev^2(X_1, X_2) \end{aligned}$$

By dividing both sides by $Dev(X_1) \cdot Dev(X_2)$ (and reverting order) we have that determinant is ≥ 0 as long as

$$\begin{aligned} &\frac{Codev(X_1, X_2)}{Dev(X_1)Dev(X_2)} \leq 1 \\ &r_{12}^2 \leq 1 \end{aligned}$$

where r_{12}^2 is the R^2, the square of correlation coefficient between the two variables r_{12},
By definition the R^2 is always ≤ 1, being the correlation coefficient in the range -1 to 1 So this is true by definition and thus $\det \mathbf{S} \geq 0$

- the determinant of its minor is non negative: here we have just 1 minor obtained by canceling the first row and first column. The minor is scalar/matrix with just $dev(x_2)$ and its determinant is just scalar itself

$$\det([dev(x_2)]) = dev(x_2) \geq 0$$

□

Important remark 8. The fact that S is positive-semidefinite, that is cannot have negative eigenvalues. have a statistical interpretation since we will se that eigenvalue are variances (that cannot be negative).
Negative eigenvalue for $\mathbf{S}$ means something gone wrong in the computation

1.1.6 Correlation matrix

The covariance between two standardized variables Z_i, Z_l is the correlation between the corresponding unstandardized ones X_i, X_l since:

$$\begin{aligned}
\text{Cov}(Z_i, Z_l) &= \frac{\sum_{j=1}^{n}(z_{ji} - \overline{z}_i)(z_{jl} - \overline{z}_l)}{n} \overset{(1)}{=} \frac{\sum_{j=1}^{n} z_{ji} - z_{jl}}{n} (= \frac{1}{n}\mathbf{z}_i^\top \mathbf{z}_l) \\
&= \frac{\sum_{j=1}^{n}(x_{ji} - \overline{x}_i)(x_{jl} - \overline{x}_l)/\sqrt{\text{Var}[X_i]\,\text{Var}[X_l]}}{n} \\
&= \frac{\sum_{j=1}^{n}(x_{ji} - \overline{x}_i)(x_{jl} - \overline{x}_l)/n}{\sqrt{\text{Var}[X_i]\,\text{Var}[X_l]}} = \frac{\text{Cov}(X_i, X_l)}{\sqrt{\text{Var}[X_i]\,\text{Var}[X_l]}} \\
&= \text{Corr}(X_i, X_l) = r_{il}
\end{aligned}$$

where in (1) considering that standardized variables has zero mean.

Definition 1.1.2. Putting all the correlations between the p variables in the correlation matrix $\mathbf{R}$ we end with:

$$\underset{p\times p}{\mathbf{R}} = \begin{bmatrix} 1 & \dots & r_{1i} & \dots & r_{1p} \\ & & 1 & \dots & r_{ip} \\ & & & & 1 \end{bmatrix}$$

where outside the main diagonal there are linear correlation coefficients between variables (-1 to 1), while on the main diagonal all are 1 (correlation between variable with itself).

Important remark 9 (Computation). To compute it, being the correlation of the unstandardized variable the covariance of the standardized ones, we can proceede as done for variance covariance matrix (but starting from the standardized variable matrix instead of the mean centered):

$$\begin{aligned}
\mathbf{R} = \text{Cov}(\mathbf{Z}) &= \frac{1}{n}\mathbf{Z}^\top\mathbf{Z} = \frac{1}{n}\mathbf{D}^{-1/2}\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\underbrace{\frac{1}{n}\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}}_{\mathbf{S}}\mathbf{D}^{-1/2} \\
&= \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}
\end{aligned}$$

it turns out that can be obtained the variance covariance matrix (numerator of corr) by both std of one and the other variables, in matrix form, just bye pre and postmultiply by $\mathbf{D}^{-1/2}$.
So we can obtain $\mathbf{R}$ both from $\mathbf{S}$ or from $\mathbf{Z}$

Important remark 10 (Properties). Since $\mathbf{R}$ is a covariance matrix it has the property of any covariance matrix so

- squared $p \times p$
- symmetric (if we transpose we get the same)
- positive semi definite

Remark 11 (Some final remarks). Either covariance matrix between standardized variables is the correlation matrix or the correlation matrix can be thaught as the covariance matrix of standardized variables.
Every time we work with $\mathbf{R}$ it's like we're working with (relationship between) standardized variables.
In study we can work either

- with $\mathbf{S}$
- with $\mathbf{R}$ if we want get rid of different measurement units

1.2 Multivariate random variables and derived linear combinations

The data matrix $\mathbf{X}$ may be thought of as describing a sequence of n empirical realizations of a p-dimensional random vector $\mathbf{x}^\top$.
In the following we will first describe multivariate statistical methods considering random vectors (i.e. at the population level) and then we will derive their sample counterpart.
Let's assume we are dealing with a p-dimensional random vector $\mathbf{x}$; we will denote

- by μ its p-dimensional expectation
- by $\mathbf{\Sigma}$ its $p \times p$ covariance matrix. It is worth remembering that, for mean centered random variables, $\mathbf{\Sigma} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]$.

Single linear combinations Most of the multivariate statistical methods we will deal with in the following are based on *linear combinations of the components of a random vector*. We will define as a single linear combination as

$$y = \mathbf{a}^\top \mathbf{x}$$

where $\mathbf{a}$ is the p-dimensional vector of coefficients. Note that y is a scalar random variable.
The expected value and the variance of a single linear combination will be:

$$\mathbb{E}[y] = \mathbb{E}\left[\mathbf{a}^\top \mathbf{x}\right] = \mathbf{a}^\top \mathbb{E}[\mathbf{x}] = \mathbf{a}^\top \mu$$
$$\mathrm{Var}[y] = \mathrm{Var}\left[\mathbf{a}^\top \mathbf{x}\right] = \mathbf{a}^\top \mathrm{Var}[\mathbf{x}]\, \mathbf{a} = \mathbf{a}^\top \mathbf{\Sigma} \mathbf{a}$$

Note that, because of the properties of the covariance matrix, the variance of a single linear combination is a *positive semi definite quadratic form.*
It is interesting to study its properties as the vector a varies: let's consider again the simple case consisting of two variables only and a generic covariance $\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix}$.
After suitably performing the scalar product we obtain:

$$\mathrm{Var}[y] = \mathbf{a}^\top \mathbf{\Sigma a} = \begin{bmatrix} a1 & a2 \end{bmatrix} \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} \begin{bmatrix} a1 \\ a2 \end{bmatrix} = a_1^2\sigma_{11} + 2a_1a_2\sigma_{12} + a_{22}\sigma_{22}$$

If we read this variance:

- as a function of a_1 we easily recognize, in the polynomial of degree 2, the equation of a parabola. The coefficient of a_1^2 is positive since it is a variance, the equation describes therefore a concave up parabola;
- the same happens if we read the variance as a function of a_2.

This means that, as a varies, $\mathrm{Var}[y]$ does never reach a finite maximum, but only a minimum.

Multiple linear combination if interested in more than one linear combination, say m, the vectors of coefficients will be the columns of a $p \times m$ matrix $\mathbf{A}$; the m linear combinations will be the components of the m-dimensional random vector obtained by

$$\mathbf{y} = \mathbf{A}^\top \mathbf{x}$$

The expected value and the variance of multiple linear combinations will be:

$$\mathbb{E}[\mathbf{y}] = \mathbb{E}\left[\mathbf{A}^\top \mathbf{x}\right] = \mathbf{A}^\top \mathbb{E}[\mathbf{x}] = \mathbf{A}^\top \mu$$
$$\mathrm{Var}[\mathbf{y}] = \mathrm{Var}\left[\mathbf{A}^\top \mathbf{x}\right] = \mathbf{A}^\top \mathrm{Var}[\mathbf{x}]\,\mathbf{A} = \mathbf{A}^\top \mathbf{\Sigma A}$$

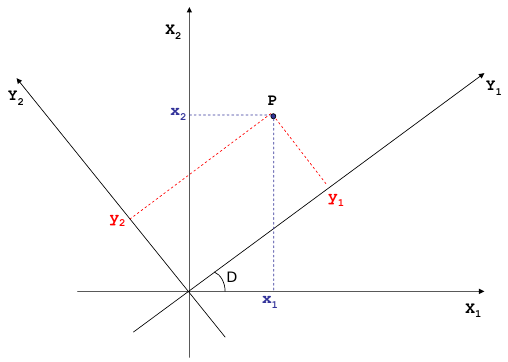
Orthogonal combinations Linear combinations defined by an *orthogonal* matrix $\mathbf{A}$ describe an axes *rotation in the multidimensional space.* A simpler two variable case may help in understanding why.
Figure 1.1 presents the coordinates of point P, both in the original reference system X_1, X_2 and in the new reference system Y_1, Y_2 obtained after rotating the system X_1, X_2 by an angle α.
The coordinates of P in the original reference system are (x_1, x_2), while in in the rotated reference system (y_1, y_2) and can be obtained from the original ones as

$$y_1 = x_1 \cos\alpha + x_2 \sin\alpha$$
$$y_2 = -x_1 \sin\alpha + x_2 \cos\alpha$$

or, with a notation coherent with the one we used before as:

$$y_1 = \mathbf{a}_1^\top \mathbf{x}$$
$$y_2 = \mathbf{a}_2^\top \mathbf{x}$$

where, because of $\sin^2\alpha + \cos^2\alpha = 1$ both $\mathbf{a}_1$ and $\mathbf{a}_2$ are unit norm vectors.
The rotated coordinates are therefore a linear combination of the original ones.



(a)

Figura 1.1: Axes rotation

Example 1.2.1. Given a bi-dimensional random vector $\mathbf{x}^\top = (x_1, x_2)$ with expected value $\mu^\top = (\mu_1, \mu_2)$ and covariance matrix

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

consider the two linear combinations

$$y_1 = x_1 - x_2$$
$$y_2 = x_1 + x_2$$

and derive the expected value and the covariance (the solution will be provided in class).

Example 1.2.2. Consider three independent standardized variables Z1 , Z2 , Z3 . Assume you transform them as follows obtaining three new variables Y1 , Y2 , Y3 :

$$Y_1 = Z_1$$
$$Y_2 = Y_1 + 0.01Z_2$$
$$Y_3 = 10Z_3$$

Derive the covariance matrix of the new Y variables.

1.3 Lab

To doing matrix manipulation let's consider the dataset `job_perf` Data contain observations on fifty police officers that were rated by their supervisors in 6 categories as part of standard police departmental administrative procedure:

- `commun`: Communication Skills

- `probl_solv`: Problem Solving
- `logical`: Logical Ability
- `learn`: Learning Ability
- `physical`: Physical Ability
- `appearance`: Appearance

To do matrix computation we convert it to matrix and extract number of units which is neede in the following

```
# matrix computation
job <- read.table("data/job_perf.txt", header = TRUE)
head(job)

##   commun probl_solv logical learn physical appearance
## 1     12         52      20    44       48         16
## 2     12         57      25    45       50         16
## 3     12         54      21    45       50         16
## 4     13         52      21    46       51         17
## 5     14         54      24    46       51         17
## 6     14         48      20    47       51         18

X <- as.matrix(job)
n <- nrow(X)
```

1.3.1 Mean centered matrix

```
C <- diag(n) - (1/n) * rep(1, n) %*% t(rep(1, n)) # mean centering matrix
Xc <- C %*% X # centered data
colMeans(Xc) ## check it's centered (mean should be 0)

##        commun    probl_solv       logical         learn      physical
## -5.639933e-16  6.679102e-15 -3.792522e-15 -9.414691e-16  1.059597e-14
##    appearance
## -2.646772e-15
```

1.3.2 Covariance matrix

```
## covariance matrix
(S <- 1/n * t(Xc) %*% Xc) ## S is 6x6

##            commun probl_solv logical  learn physical appearance
## commun     7.3776     0.8512  1.5064 7.4896   6.3312     7.7992
## probl_solv 0.8512     5.6944  2.1368 0.9552   0.9344     0.9104
## logical    1.5064     2.1368  6.0596 1.5944   1.5168     1.5788
## learn      7.4896     0.9552  1.5944 7.8816   6.5752     8.0832
```

```
## physical   6.3312     0.9344  1.5168 6.5752   5.6944     6.8504
## appearance 7.7992     0.9104  1.5788 8.0832   6.8504     8.7764

## using cov we get the unbiased covariance so to obtain the same..
cov(X) * (n-1) / n

##            commun probl_solv logical  learn physical appearance
## commun     7.3776     0.8512  1.5064 7.4896   6.3312     7.7992
## probl_solv 0.8512     5.6944  2.1368 0.9552   0.9344     0.9104
## logical    1.5064     2.1368  6.0596 1.5944   1.5168     1.5788
## learn      7.4896     0.9552  1.5944 7.8816   6.5752     8.0832
## physical   6.3312     0.9344  1.5168 6.5752   5.6944     6.8504
## appearance 7.7992     0.9104  1.5788 8.0832   6.8504     8.7764
```

1.3.3 Standardized data

```
## for the standardized data: D is diagonal matrix that has variances
D <- diag(diag(S))
# diag(S) only extracts the diagonal of our matrix (which has the variances):
# we use another diag to make a diagonal matrix

## do do the ^{-1/2}  we have to do solve which correspond to -1
Z <- Xc %*% solve(D^0.5)

## check that standardized (mean = 1, sd = var = 1)
round(colMeans(Z), 5) # all 0

## [1] 0 0 0 0 0 0

apply(Z, 2, var) # more or less all standardized

## [1] 1.020408 1.020408 1.020408 1.020408 1.020408 1.020408
```

1.3.4 Correlation matrix

```
## can be found in different ways (covaraince of standardized data or starting
## from the covaraince matrix)

(R <- 1/n * t(Z) %*% Z) #cov: on diagonal 1 (it's a corr)

##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 1.0000000 0.1313258 0.2252998 0.9821863 0.9767974 0.9692466
## [2,] 0.1313258 1.0000000 0.3637625 0.1425815 0.1640910 0.1287805
## [3,] 0.2252998 0.3637625 1.0000000 0.2307109 0.2582155 0.2164945
## [4,] 0.9821863 0.1425815 0.2307109 1.0000000 0.9814717 0.9718918
## [5,] 0.9767974 0.1640910 0.2582155 0.9814717 1.0000000 0.9690222
## [6,] 0.9692466 0.1287805 0.2164945 0.9718918 0.9690222 1.0000000
```

```r
(R <- solve(D^0.5) %*% S %*% solve(D^0.5)) ## same results obtained with D^{-1/2} S D^{-1/2}
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 1.0000000 0.1313258 0.2252998 0.9821863 0.9767974 0.9692466
## [2,] 0.1313258 1.0000000 0.3637625 0.1425815 0.1640910 0.1287805
## [3,] 0.2252998 0.3637625 1.0000000 0.2307109 0.2582155 0.2164945
## [4,] 0.9821863 0.1425815 0.2307109 1.0000000 0.9814717 0.9718918
## [5,] 0.9767974 0.1640910 0.2582155 0.9814717 1.0000000 0.9690222
## [6,] 0.9692466 0.1287805 0.2164945 0.9718918 0.9690222 1.0000000
```

```r
cor(X) ## same again
```

```
##               commun probl_solv   logical     learn  physical appearance
## commun     1.0000000  0.1313258 0.2252998 0.9821863 0.9767974  0.9692466
## probl_solv 0.1313258  1.0000000 0.3637625 0.1425815 0.1640910  0.1287805
## logical    0.2252998  0.3637625 1.0000000 0.2307109 0.2582155  0.2164945
## learn      0.9821863  0.1425815 0.2307109 1.0000000 0.9814717  0.9718918
## physical   0.9767974  0.1640910 0.2582155 0.9814717 1.0000000  0.9690222
## appearance 0.9692466  0.1287805 0.2164945 0.9718918 0.9690222  1.0000000
```

Capitolo 2

Cluster analysis

So far we focused on the columns (variances mean covariances). Now we start to see the dataset row-wise.
Clustering a set of n objects into k groups is usually moved by the aim of identifying internally homogenous groups according to a specific set of variables. In order to accomplish this objective, the starting point is computing a matrix, called dissimilarity matrix, which contains information about the dissimilarity of the observed units.
Our focus is to measure similarity or dissimilarity of units; we want to read the data matrix row-wise and compare each unit to each other units.
According to the nature of the observed variables (quantitative, qualitative, binary or mixed type variables), we can define and use different measures of dissimilarity. We need to define some metrics and matrices before proceeding.

Remark 12. Disclaimer: we have more object that letters in our alphabet sometimes we use the same letter but the context will clarify. Here the matrix $\mathbf{D}$ will be used as distance matrix (previously used as diagonal matrix).

2.1 Dissimilarity measures

Let's consider the $n \times p$ data matrix, where n is the number of observed units and p is the number of observed variables. Let's indicate with i and j two generic objects and with x_{ih} the value of the h-th variable for the i-th object.

Definition 2.1.1 (Distance/dissimarity matrix)**.** The distance or dissimilarity matrix $\mathbf{D}$ is $n \times n$ and comprises all the distances/dissimilarity between units belonging to the sample.

Remark 13. In most cases it is symmetric (if we use distance, which are symmetric) but not need to be so.

Important remark 11. Let's consider distance function between units. The various type depends on the kind of data they apply

2.1.1 All numeric variables

Remark 14. If the observed variables are all quantitative, each unit can be identified with a point in the p-dimensional space.

Example 2.1.1. In two dimensions

	height	weight
vadim	182	80
daulet	185	87

data can be represented as point in the two dimensional space.
With tree variables we have points in space, with p-variable the unit are point in the p-dimensional variable.

Remark 15. Since data are mapped to points I can measure the difference between units as distance between corresponding points; the more different they are the more distant the corresponding will be, while if the points are overlapping the distance will be 0.

The dissimilarity between two objects i and j in this space (of all quantitative variables) can be measured through:

- **euclidean distance**: in the 2-dimensional space is the length of the segment joining the two points (just applying pitagora's theorem). In general the distance between unit j and h is

$$d_{ij} = \sqrt{\sum_{h=1}^{p}(x_{ih} - x_{jh})^2}$$

  which is also called the L_2-norm

- city-block/Manhattan distance:

$$d_{ij} = \sum_{h=1}^{p}|x_{ih} - x_{jh}|$$

  The name comes from the fact that in Manhattan we can't walk diagonal so we need to count the segments of streets which separates two point. Also called L_1-norm

- **Minkowski distance**: the previous are specialization of the following family of distances

$$d_{ij} = \sqrt[m]{\sum_{h=1}^{p}|x_{ih} - x_{jh}|^m}$$

  If $m = 1$ we obtain the city block distance, while if $m = 2$ we obtain the Euclidean distance.
  In general larger and values for m put larger weights on large differences (difference is large and raise to a large power take power on the other differences on other variables).

Remark 16. For all of the three distances the properties of a distance function hold:

$$\begin{aligned} &d_{ij} \geq 0 \\ &d_{ii} = 0 \\ &d_{ij} = d_{ji} \\ &d_{ij} \leq d_{it} + d_{tj} \quad \text{triangle inequality} \end{aligned}$$

V D	1	0	
1	a	b	
0	c	d	
			p

Tabella 2.1: The mf counts table

Example 2.1.2. To see the impact of increasing m on variables relevance see

manhattan: $d_{vd} = |182 - 185| + |80 - 87| = 3 + 7 = 10$

euclidean: $d_{vd} = \sqrt{(182 - 185)^2 + (80 - 87)^2} = \sqrt{9 + 49} = \sqrt{58} = 7.62$

Going from Manhattan to Euclidean distance become smaller but the importance of weight variable (were the difference between units is higher) become bigger

2.1.2 All binary variables

Remark 17. When dealing with binary or categorical variables it is not possible to define true distance measures, but one can use some dissimilarity measures which:

- are still close to zero, when i and j are similar to each other
- increase when differences between two units are larger
- are symmetric
- are equal to zero when considering the dissimilarity of an object with itself

Nevertheless, these dissimilarity measures generally do not satisfy the triangle inequality.

Example 2.1.3. Considering that xx_{ih} can assume only two values, coded as 0 and 1, we can have a dataset like this

```
        brother  smoke  likebeer likechocolate
vadim         1      0         1              0
daulet        1      0         0              1
```

How to measure how units are similar with respect to p binary variables. A table such as 2.1 is constructed with the count of variable where the two units have agreement or disagreement where

- a: number of times (variables) where units agree on 1
- d: number of times the units agree on 0
- b: n of times unit i shows the 1 while the unit h show 0 (disagree)
- c: n of times unit i shows the 0 while the unit h show 1 (disagree)

All sum of all the entries will be p, the number of variable. In the example above we have $a = b = c = d = 1$. Basing ourselves on the matrix we have several dissimilarity coefficients:

1. *simple dissimilarity coefficient*

$$d_{ij} = \frac{\text{n. disagree}}{n.\ variables} = \frac{b+c}{a+b+c+d}$$
$$= 1 - s_{ij} = 1 - \frac{a+d}{a+b+c+d}$$

where s_{ih} is called simple similarity coefficient.
It's called *dissimilarity* and not *distance* because a distance function requires to be positive symmetric AND to satisfy triangular inequality; this function does not satisfy the third property (dissimilarity usually don't)

2. *Jaccard coefficient*: is defined as

$$d_{ij} = \frac{a}{a+b+c}$$

It is recommended when binary variables indicate presence/absence of a specific characteristic.[1]
In this case agreement 1/1 can have a different meaning than agreement 0/0; indeed, absence of a certain characteristic in both objects i and j does not contribute to improve the similarity between them. Hence, it can be more useful to divide by $a+b+c$ rather than by $a+b+c+d$.
Generalizing, Jaccard consider in many cases the agreement on null non necessarily means similarity (thus ignore/eliminates in the numerator and denominator the agreement on null).

Example 2.1.4. In our example, with $a = b = c = d$ we have that the simple dissimilarity between vadim and daulet is

$$d_{vd} = \frac{2}{4} = 0.5$$

so the two units are identical with respect to half o characteristics considered.

2.1.3 All categorical variables

Example 2.1.5. Suppose

```
        phone   fav.team  zodiac blood
vadim  iphone barcellona  gemini    A-
daulet iphone juventus      lion    B+
```

When observed variables are categorical with more than two levels, a simple measure of the degree of dissimilarity between two units is given by the number/percentage of disagreeing of variables

$$d_{ij} = 1 - \frac{\text{n agreement}}{\text{n variables}} = \frac{\text{n disagreement}}{p}$$

However this is a poor measure of similarity which does not take account the number of levels of each variables (eg is easy to have distance in zodiac sign more easy in the phone iphone vs android)

[1] These measures were defined for taxonomy studies and the used to compare different animals. In these cases there are many agreement on not having certain characteristics. But having agreement on not having wings (cows and snake) does not necessarily mean that two animals (units) are similar.

2.1.4 Mixed types

Example 2.1.6.

	height	weight	phone	brother	beer
vadim	182	80	iphone	1	1
daulet	185	87	iphone	1	0

A popular measure here is the Gower's dissimilarity coefficient which is defined as the complement of similarity measure

$$d_{ij} = 1 - s_{ij}$$
$$s_{ij} = \frac{\sum_{h=1}^{p} \delta_{ijh} s_{ijh}}{\sum_{h=1}^{p} \delta_{ijh}}$$

where

- δ_{ijh} is a simply way to taking into account missingness, and defined as

  $$\delta_{ijh} = \begin{cases} 1 & \text{if comparison between unit } i \text{ and } j \text{ is possible for variable } h \\ 0 & \text{otherwise} \end{cases}$$

  If we have missing values the two units cannot be compared and in this case it is impossible to use data to tell difference. If all the variables can be compared at the numerator we have p, which is a counter where both the units are non-missing.

- the similarity s_{ijh} quantity is defined separatedly for *each variable*. If the variable is:

  – numeric

  $$s_{ijh} = 1 - \frac{|x_{ih} - x_{jh}|}{\text{range of } x_h} = 1 - \frac{|x_{ih} - x_{jh}|}{\max(x_h) - \min(x_h)}$$

  – binary

  $$s_{ijh} = \begin{cases} 1 & \text{the units } i \text{ and } j \text{ agree wrt variable } h \\ 0 & \text{otherwise} \end{cases}$$

Example 2.1.7. Eg for height which is numerical

$$s_{vd,height} = 1 - \frac{|182 - 185|}{190 - 160} = 1 - \frac{3}{30} = \frac{9}{10}$$

where 190 and 160 were the height of taller and smaller in class. So being 9/10 the two units are very similar (close to 1).
In case of considering the similarity of the most distant units we have

$$s_{vd,height} = 1 - \frac{|190 - 160|}{190 - 160} = 0$$

and we have max difference.
For what concerns the similarity in weight

$$s_{vd,weight} = 1 - \frac{|87 - 80|}{90 - 48} = 1 - \frac{7}{46} = \frac{5}{6} = 0.83$$

where 90 and 48 are the weights of the fatter and thinner in class. So 0.83 is still highly similar units.
Thus finally overall we have that the Gower dissimilarity is

$$d_{dv} = 1 - s_{dv} = 1 - \frac{0.9 + 0.83 + 1 + 0}{4} = 1 - 0.68 = 0.32$$

Dissimilarity between the two units is 0.32 and we do the same things for all the units

2.2 Clustering

Some facts:

- A *cluster* is a group of unit.
- *Cluster analysis* is composed by a very broad sets of techniques: one can define groups in many way and thus obtain many different solutions.
- The common *goal* of this analysis to find groups of units that are similar within group and separate/different between as much as we can (wrt the variable we take into account).
- Important *starting point* in variable selection: we have to choose carefully the charateristics to analyze (same set of units can provide very different information).

Traditionally two type of methods are used:

- *hierarchical methods*: the methods work by building a series of *nested* classification (small group inside larger and so on), represented by *dendrograms*.
  By cutting the graph at a certain level of dissimilarity we obtain a partition of objects into different groups. Each partition in k groups is optimal in a step-wise sense. It is the best partition in k group one can obtain, given the partition obtained at the previous step.
  Hierarchical methods can be divided in two family (based on the way on which is builded):
  - *agglomerative methods* (bottom-up) (which is the one we will study): start from n cluster group each composed by 1 unit, and we put clusters together/merge them until we obtain the big group.
    Notice that for agglomerative hierarchical clustering methods, when two clusters are merged, they will remain so until the end of the hierarchy; a mistake committed in the initial phase of classification can never be corrected. This is both the strength and the weakness of this kind of methods; the bigger is the number of units the higher is the probability of misclassifying a unit.
    If the aim is to find a number of clusters much lower than the number of units ($k << n$), divisive methods have a lower risk to commit a mistake, since the number of steps needed to reach partition in k groups is smaller (which however has the drawback of being computational complex);

  - *divisive methods* (top-down): splitting in subgroups up to when each group has 1 unit.
    Most famous divisive methods is due to Edwards and Cavalli Sforza (biological domani, numerical variables only based onthe decomposition of total sum of square in the within/between, in order to have the smallest within and larger between). One goes on up to when variability is zero in each group ($n = 1$ for each).
    This approach is not very much used because very computational demanding (we have to try each splitting) and furthermore only for numerical variabile. Finally if we make mistakes in the first stages, the error goes on and on (contrary to agglomerative)

- *partitioning methods*: clustering method that produces a single-step *flat* partition in k-groups; it's not an hierarchy anymore, its a flat partition.

Definition 2.2.1 (Singleton)**.** A group/cluster of just 1 unit

2.3 Hierarchical agglomerative methods

These methods:

1. initially consider each unit as a separate cluster and calculate distance between cluster (as simple distance between units). Regarding distances we can apply any metric which is appropriate for types of data at hand;

2. they subsequently merge in one cluster those units that have the lowest dissimilarity; if we have two *identical minimum distances*, we toss a coin to choose which one to aggregate;

3. then, they evaluate dissimilarities between this new cluster and the others, merging again the two groups that have the lowest value in the dissimilarity matrix.
   Clustering methods within this family are classified according how *dissimilarity between groups/clusters* is defined This is defined as:

   (a) *single linkage*: minimum distance between units belonging to different cluster (least distant couple);

   (b) *complete linkage*: maximum distance between units belonging to different cluster (most distant couple of units);

   (c) *agerage linkage*: mean distance between units belonging to different cluster (compromise of the two above).

   In case of *numerical variables only* we have two more methods:

   (a) centroid methods

   (b) Ward's methods

Remark 18. Sometimes all these methods (even centroid and Ward's) are listed among linkage methods because two statisticians, Lance and Williams put all the methods seen so far in a general family of clustering methods (that depends on various parameters).

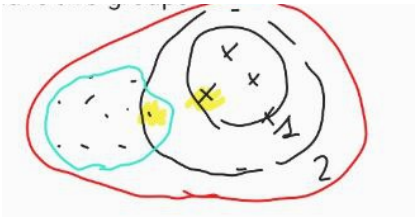


Figura 2.1: chaining effect

2.3.1 Single linkage (nearest neighbor

Definition 2.3.1. Having two groups C and G, their distance is the minimum distance between the units

$$d_{CG} = \min_{i \in C, j \in G} d_{ij}$$

Remark 19. Thus one has to compute all the pairwise distances between all the units in the first and the second group: then the distance between the groups is the distance between the closest point belonging to the different groups.

Important remark 12. Clusters obtained with the Single Linkage method tend to have a long and narrow shape and to be internally poorly homogenous ("strange groups" since has the tendency to put togheter heterogenous units), given that only two similar observations from two different clusters are enough in order to merge the two clusters (*chaining effect*, fig 2.1).
For this reason, this method is not able to identify overlapping clusters, but it is good when clusters look well separated, because in every cluster yielded by the procedure any unit is more similar to the units belonging to the same cluster than to units of different groups.

Example 2.3.1. Assume we have thid distance matrix $\mathbf{D}$ between units of our sample (a, b, c, e) per il momento indico in questo modo la matrice facendo riferimento alle unità ...

$$\mathbf{D} = \begin{bmatrix} & a & b & c & e \\ a & 0 & 1 & 4 & 3 \\ b & & 0 & 2 & 7 \\ c & & & 0 & 5 \\ e & & & & 0 \end{bmatrix}$$

To cluster this dataset using single link we start by putting together the closest units: a and b are the two most similar which have **distance 1**. To update the distance matrix and re-iterate, we apply the single linkage definition to calculate the distance of these groups. We obtain

$$\mathbf{D} = \begin{bmatrix} & (a,b) & c & e \\ (a,b) & 0 & 2 & 3 \\ c & & 0 & 5 \\ e & & & 0 \end{bmatrix}$$

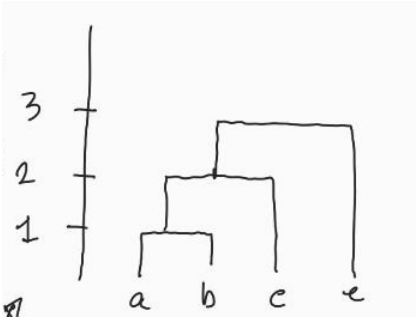


Figura 2.2: dendo1

where the elements of the new first row (the only changing) were calculated as

$$d_{(a,b),c} = \min(d_{ac}, d_{bc}) = \min(4, 2) = 2$$
$$d_{(a,b),e} = \min(d_{ae}, d_{be}) = \min(3, 7) = 3$$

Now we iterate and choose to aggregate the group with the minimum distance which are (a, b) and c with **distance 2**. The new distance will be

$$\mathbf{D} = \begin{bmatrix} & (a,b,c) & e \\ (a,b,c) & 0 & 3 \\ e & & 0 \end{bmatrix}$$

where we obtained

$$d_{(a,b,c),e} = \min(d_{ae}, d_{be}, d_{ce}) = \min(3, 5, 7) = 3$$

So the final distance between the units will be **3**.
To build the dendogram, we put units on the x axis and distances on which are grouped on the y axis (fig 2.2)

2.3.2 Complete linkage (furthest neighbor)

Definition 2.3.2. Distance between two clusters C and G is defined as the dissimilarity between the two furthest-away points:

$$d_{CM} = \max_{i \in C, j \in G} d_{ij}$$

Remark 20. i.e. it is the longest distance between any element i of C and any element j of G. In other words, d_{CG} is the diameter of the smallest sphere that can wrap the cluster obtained by merging C and G.

Important remark 13. Complete linkage tends to produce *spherical/compact grops* it splits data into balls; when we have big jump it means that we're trying to put together groups which are more different. Nothing can be said about their separation.

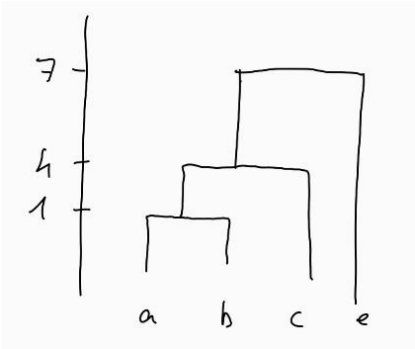


Figura 2.3: dendo2

Example 2.3.2. Starting from the very same matrix of differences

$$\mathbf{D} = \begin{bmatrix} & a & b & c & e \\ a & 0 & 1 & 4 & 3 \\ b & & 0 & 2 & 7 \\ c & & & 0 & 5 \\ e & & & & 0 \end{bmatrix}$$

we put together the most similar units, so a, b (which have **distance 1**); the only change is in how we calculate the distance between groups and update the distance matrix

$$\mathbf{D} = \begin{bmatrix} & (a,b) & c & e \\ (a,b) & 0 & 4 & 7 \\ c & & 0 & 5 \\ e & & & 0 \end{bmatrix}$$

where we calculated

$$d_{(a,b),c} = \max d_{ac}, d_{bc} = \max 4, 2 = 4$$
$$d_{(a,b),e} = \max d_{ae}, d_{be} = \max 3, 7 = 7$$

Now to aggregate I look for the smallest value, which as before (not always the case, btw) are between (a, b) and c with **distance 4**. We construct the final matrix as

$$\mathbf{D} = \begin{bmatrix} & (a,b,c) & e \\ (a,b,c) & 0 & 7 \\ e & & 0 \end{bmatrix}$$

where

$$d_{(a,b,c),e} = \max d_{ae}, d_{be}, d_{ce} = \max 3, 5, 7 = 7$$

So finally we put all the units together with **distance 7** (fig 2.3)

2.3.3 Average linkage

Definition 2.3.3. Distance between two clusters C and G is defined as the average of all the dissimilarities between pairs of points one from each cluster

$$d_{CG} = \frac{\sum_{i \in C, j \in G} d_{ij}}{n_c \cdot n_g}$$

where n_C is the number of elements in cluster C, and n_G is the number of elements in cluster G, and at the denominator we have all the comparisons between units of the two groups.

Example 2.3.3. Starting from the very same matrix

$$\mathbf{D} = \begin{bmatrix} & a & b & c & e \\ a & 0 & 1 & 4 & 3 \\ b & & 0 & 2 & 7 \\ c & & & 0 & 5 \\ e & & & & 0 \end{bmatrix}$$

After the first aggregation

$$\mathbf{D} = \begin{bmatrix} & (a,b) & c & e \\ (a,b) & 0 & 3 & 5 \\ c & & 0 & 5 \\ e & & & 0 \end{bmatrix}$$

where we obtained

$$d_{(a,b),c} = \frac{1}{2 \cdot 1}(4+2) = 3$$
$$d_{(a,b),e} = \frac{1}{2 \cdot 1}(3+7) = 5$$

The second step is to merge unit c with group (a,b) (having **distance 3**) obtaining

$$\mathbf{D} = \begin{bmatrix} & (a,b,c) & e \\ (a,b,c) & 0 & 5 \\ e & & 0 \end{bmatrix}$$

where

$$d_{(a,cb,c),e} = \frac{1}{3 \cdot 1}(3+7+5) = 5$$

The final one put together the remaining, having distance 5 (fig 2.4)

2.3.4 Centroid methods

Important remark 14. This and the following methods are for numeric variables only.

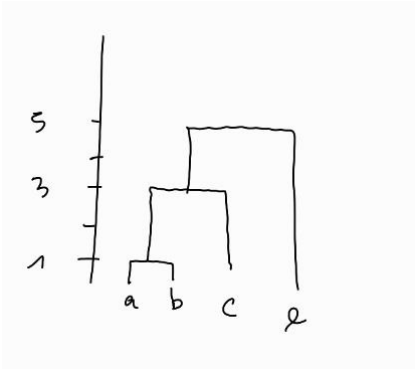


Figura 2.4: dendo3

Definition 2.3.4. Distance between two clusters C and G is defined as the Euclidean distance between the two baricentres:

$$d_{CG} = \|\overline{\mathbf{x}_C} - \overline{\mathbf{x}_G}\|$$

where $\overline{\mathbf{x}_C}$ is the mean vector of C and $\overline{\mathbf{x}_G}$ is the mean vector of G

Remark 21. That is these are the mid points of the cloud of points associated to a given group in the space; and the distance between them is the distance between the cluster they represent.
Obviously we need numeric variables only to compute the mean vectors; in principle we would need the data matrix (to compute mean vector every time).

Remark 22. A problem that may be encountered with this approach is that dissimilarity of a union is lower than the one measured at the previous merge (because of the centroid shift) and this makes the results less interpretable

2.3.5 Ward's method

Remark 23. This is the agglomerative analog of Edward-Cavalli-Sforza divisive method. It's based on the decomposition of total sum of square in the within and between components.

Definition 2.3.5. Distance between two clusters C and G is defined as the deviance between the two clusters. Be x_{ihs} the value of variable h observed on unit i in cluster s and be $\mathbf{x}_{hs}$ the average of variable h in cluster s; then

$$E_s = \sum_{h=1}^{p} \sum_{i=1}^{n_s} (x_{ihs} - \mathbf{x}_{hs})^2$$

E_s is the deviance within s-th clister and $E = \sum_s E_s$ the deviance within the whole partition (within sum of square, WSS).

Important remark 15. The process

- as usual the classification process starts by considering n cluster each composed by 1 unit: thus our within group sum of WSS=0 (single unit identical to themselves);

- each merge leads to an increase of the 'deviance within' a cluster, an increase in WSS: the algorithm looks for the merge that will imply the smallest increase of the 'deviance within' the whole partition (create groups which are most similar)

Remark 24. Some remarks:

- since we work with sum of square it's obvious that we need numerical variables, like the centroid linkage;
- differently from centroid, it does not have problem with the order of dissimilarities in the different levels of the hierarchy;
- both methods identify spherical clusters.
- this was the first method proposed in statistical literature
- has have no strong math properties but they're very populare.
- can be slow

2.4 Partitioning methods

Remark 25. Differently from hierarchical methods, partitive methods give rise to a single partition with k groups, where k is previously specified or is determined by the clustering method itself. Data are split into k clusters in such a way the within-cluster 'dissimilarity' is minimized.
The problem seems simple, but actually the numbers involved make impossible to consider all possible partition of n individuals into k groups. That is why algorithms designed to search for the minimum values of the clustering criterion, have then been developed.

Remark 26. The most famous methods among the partitioning methods is k-means clustering; as the name says the methods *works for numerics variable only.*

2.4.1 k-means clustering

We want to obtain the partition in k groups, with k given, having the smallest WSS; having our dataset and considered a given k eg $k = 3$.

Remark 27 (One k-means algorithm). One of the methods (not the unique) with which to implement k-means is the following

```
|----------------------| x <- consider first three rows
|----------------------| x <- of our dataset
|----------------------| x <- as starting centres
|----------------------|
...
|----------------------|
|----------------------|
```

The algorithm start choosing the first three rows in the dataset and consider them as centers of the $k = 3$ groups we want. Then:

- i start reading my data matrix and assign the fourth unit to the center to which it is closer. eg i assign unit 4 to the second center;
- i compute the center again and I continue reading the matrix in the same way, recomputing the centres after each assignment;
- once processed all the dataset we have obtained some reasonable (three) centers; then we start reading again the dataset and reassign all the observation to the nearest center, that is to produce the final assignment

Remark 28. The method:

- seems time consuming but it's faster than hierarchical methods; procedure is called `fastclust` in SAS and allows to analyze a large number of units at the same time;
- is also known as *dynamic cloud* because every time a unit is added the cloud, its cloud/means moves (and the centre as well)

Remark 29. Problem with k-means is that one has to chose k in advance: we have to have a criterion how many groups to choose,

2.4.1.1 Choice of k

Remark 30. To choose what's the most reasonable value for k we need to measure quality of clustering: two different measures are silhouette scores and Pearson's γ

Silhouette score For a partition of n units into k clusters $C_1, \ldots, C_k$, suppose unit i has been assigned to cluster C_h. We indicate with:

- $a(i)$ be the average dissimilarity between unit i (belonging to cluster h-th C_h) to all the other units of cluster C_h:

$$a(i,h) = \frac{1}{|C_h| - 1} \sum_{j \in C_h, j \neq i} d(i,j)$$

- $d(i, C_l)$ the average dissimilarity between unit i and all units of cluster C_l (different from C_h)

$$d(i, C_l) = \frac{1}{|C_l|} \sum_{j \in C_l} d(i,j)$$

  we compute $d(i, C_l)$ for all clusters C_l different from C_h to which i belongs;

- $b(i)$ the smallest $d(i, C_l)$ for unit i:

$$b(i) = \min_{i \notin C_l} d(i, C_l)$$

  The cluster for which this minimum is obtained is call neighbour of object i; this is like the second-best choice for object i.

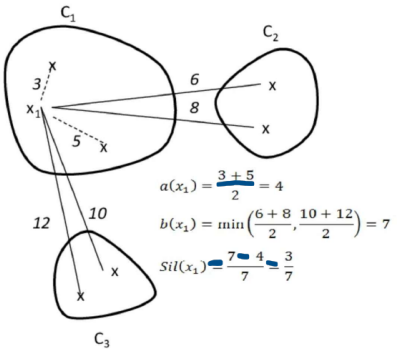


Figura 2.5: silhouette calc

- $s(i)$, silhouette for the unit i, is obtained by combining $a(i)$ and $b(i)$ as follows

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \end{cases} = \frac{b(i) - a(i)}{\max[a(j), b(j)]}$$

  in the last I expect that the numerator is positive (more similar to the groups the units belongs to respect to other), then I normalize the quantity by considering the max value.
  It can be easily seen that $-1 \leq s(i) \leq 1$. When

  - $s(i)$ is at its largest (that is, close to 1), the i-th units have been perfectly assigned: the 'within' dissimilarity $a(i)$ is much smaller than the smallest 'between' dissimilarity $b(i)$. The second best choice is not nearly as close at the actual choice.
  - $s(i)$ is about zero, then $a(i)$ and $b(i)$ are approximately equal and so it is not clear whether i should have been assigned to C_h or to C_l, it lies equally far away from both; i could be assigned equally to both (in some sense assignment done by clustering is neutral, one group or the other is the same)
  - $s(i)$ is close to -1 we have had a bad assignment for i: $a(i)$ is actually much larger than $b(i)$, and hence i lies on average closer to C_l than to C_h; therefore it would have seemed better to assign object i to its neighbour, the second best group wold have been better.

  The silhouette $s(i)$ hence measures how well unit i has been classified.

Example 2.4.1. In the example (fig 2.5) the silhouette of x_1

$$a(x_1) = \frac{3+5}{2} = 4$$
$$b(x_1) = \min\left(\frac{6+8}{2}, \frac{10+12}{2}\right) = 7$$
$$s(x_1) = \frac{7-4}{7} = \frac{3}{7}$$

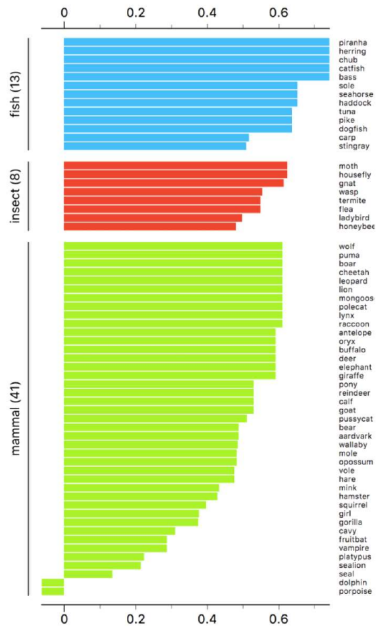


Figura 2.6: silhouette plot

So if $s(i)$ is larger than 0, as in this case, than this is good assignment.

Important remark 16. Individual silhouette are usually plotted in the *silhouette plot* (fig 2.6): graph of the silhouette profile of each individual is reported and coloring according to the group (meaningful if we have few units).

Important remark 17. To measure the quality a clustering solution we compute the **average silhouette width** (ASW)

$$\overline{s} = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

Some units will be correclty classified, some neutral, some other badly classified; we expect that the clustering is better if $\overline{s}$ is near its maximum, 1. We choose k by maximizing this statistics;

Remark 31. Mean silhouette expression

- leads to a clustering solution that emphasises the separation between the clusters and their neighbouring clusters.
- Drawback: we have to have at least two groups, so this method does not allow to check if actually the best k is 1, that is it doesn't allow to check the actual existence of different groups.

Pearson's Gamma

Remark 32. Another measure of clustering quality is known as Pearson's Gamma, written k_Γ: this is nothing but a linear correlation coefficient r, computed

on special things.
It maximises the Pearson correlation $\rho(\mathbf{d}, \mathbf{m})$ between the vector $\mathbf{d}$ of pairwise dissimilarities and the binary vector $\mathbf{m}$ that is 0 for every pair of observations in the same cluster and 1 for every pair of observations in different clusters.

Example 2.4.2. In the example we've put together a, b, c and let e alone. To compute Pearson's gamma (single linkage example)

- we start from the original dissimilarity matrix

$$\mathbf{D} = \begin{bmatrix} & a & b & c & e \\ a & 0 & 1 & 4 & 3 \\ b & & 0 & 2 & 7 \\ c & & & 0 & 5 \\ e & & & & 0 \end{bmatrix}$$

- we "vectorize" the matrix $\mathbf{D}$ into $\mathbf{d}$ putting all upper triangular distances (outside diagonal), and add a dummy vector/variable $\mathbf{g}$ which takes value 1 if units after clustering are assigned to different group, 0 otherwise

$$\mathbf{d} = \begin{bmatrix} d_{ab} \\ d_{ac} \\ d_{ae} \\ d_{bc} \\ d_{be} \\ d_{ce} \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ 2 \\ 7 \\ 5 \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} g_{ab} \\ g_{ac} \\ g_{ae} \\ g_{bc} \\ g_{be} \\ g_{ce} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

  In the final clustering we had we had a, b, c and e alone; above in g is reported the corresponding vector.

- Pearson's Gamma is just the correlation coefficient between the distance $\mathbf{d}$ and and the fact that units are assigned to different groups or the same group $\mathbf{g}$; explicitly

$$\begin{aligned} k_\Gamma &= \frac{\sum_{i=1}^{n(n-1)/2} d_i g_i - \frac{n(n-1)}{2}\overline{d}\overline{g}}{\sqrt{\sum_{i=1}^{n(n-1)/2} d_i^2 - \frac{n(n-1)}{2}\overline{d}^2}\sqrt{\sum_{i=1}^{n(n-1)/2} g_i^2 - \frac{n(n-1)}{2}\overline{g}^2}} \\ &= \frac{codev(\mathbf{d}, \mathbf{f})}{\sqrt{dev(\mathbf{d})dev(\mathbf{g})}} \end{aligned}$$

  Note that the sum is on i which are the number of unique elements out of the diagonal.

Important remark 18. The measure is a correlation which take range between -1 and 1 as usual and:

- large *positive values* close to +1 means that distant units tend to be clustered in different group and close units tend to be clustered together;
- *values close to 0* means that the classification is almost random;
- *negative values* mean that the classification is a mess: we have put in the same group different units and put in different groups similar units.

So again as before the larger the better: good clustering correspond to large positive values for k_Γ.

2.4.2 Other methods

We briefly see two methods which are becoming more and more popular in clustering literature.

2.4.2.1 PAM

Close to k-means, PAM[2] stands *Partitioning Around Medoids*: a medoid is a prototypical unit, an actual/observed unit which act as center.
The goal is to find the unit that best represents the groups; algorithmically it becomes more complicate than k-means but allows to work with non numeric variables (we can use any distance that we've seen and find the unit which minimise that distances).

2.4.2.2 Model based clustering

Remark 33. All methods seen so far are distribution free, no assumption were made regarding probability distribution generating data at hand.
On the other hand if we can rely on probabilistic assumptions there's a big family of method, the *model based clustering*, which rely on the hypothesis that data are generated by a probabilistic model.
Idea is not new (Pearson did it in univariate case)

Assume we want study height, so X is height with $F(x)$ its pdf. After having obtained the data we can imagine that height of the group is generated by sampling from a probability density.
But the point is that in this group there're males and females; so overall F is a mixture of two probability functions/density, from males and density of females:

$$F(x) = \pi_m F_m(x) + \pi_f F_f(x)$$

where π_m and π_f are proportion of males and females (*mixing proportions*), and we assume $\pi_m + \pi_f = 1$ in the population. Now I can make assumption

$$F_m(X) \sim \mathrm{N}\left(\mu_m, \sigma_m^2\right)$$
$$F_f(X) \sim \mathrm{N}\left(\mu_f, \sigma_f^2\right)$$

and rewrite my $F(x)$ as

$$F(x) = \pi_m \cdot \phi(x, \mu_m, \sigma_m^2) + \pi_f \cdot \phi(x, \mu_f, \sigma_f^2)$$

where ϕ is probability density function of normal distribution computed in x, with mean and variance coming from the two population. This above is called a *finite mixture model*: in order to obtain the observed and unknown $F(x)$ we need to guess $\mu_m, \mu_f, \pi_m, \pi_f, \sigma_f^2, \sigma_m^2$.
Pearson started from a simpler case: assume we *know* the mixing proportions π_m, π_f, and that the shape of my component densities is Gaussian. My problem

[2]Peter Rousseaw has invented this method and created the Rousseaw prize (nobel prize for statistics). First winner were research from Harvard on causal inference. Second those the FALSE discovery rates (benjamini hochberg)

is to estimate the parameters of the normal distribution which gives me the $F(x)$ that best fit my data[3]

$$F(\mathbf{x}) = \sum_{g=1}^{G} \pi_g \cdot F_g(\mathbf{x}) = \sum_{g=1}^{G} \pi_g \cdot \phi(\mathbf{x}, \mu_g, \Sigma_g)$$

with prior probability of group membership $\sum_{g=1}^{G} \pi_g = 1$ and F_g groups specific density functions; this is called gaussian mixture models (GMM).
A priori we don't have number of groups but we have the likelihood to guide us to estimate to select good value for parameters.

Remark 34 (Concluding remarks). k-means has been proved to be one particular case of these model where variables are uncorrelated and have the same variance (variance covariance matrix is diagonal).
Other methods have been proposed which are not based on gaussian (mixture of multivariate T, and so on).

2.5 Lab

Clustering a set of n objects into k groups is usually moved by the aim of identifying internally homogeneous groups according to a specific set of variables. In order to accomplish this objective, the starting point is computing a matrix, called dissimilarity matrix, which contains information about the dissimilarity of the observed units. According to the nature of the observed variables (quantitative, qualitative, binary or mixed type variables), we can define and use different measures of dissimilarity.

2.5.1 Example 1 - distance on binary data

The dataset `data.txt` contains information about the presence (1) or the absence (0) of a given attribute on each unit.

```
(data <- read.table("data/data.txt", header = TRUE))

##   Att.1 Att.2 Att.3 Att.4 Att.5 Att.6
## 1     1     1     1     0     0     1
## 2     1     0     0     1     0     1
## 3     1     0     0     1     0     1
## 4     0     0     0     0     1     0

if (FALSE) (data <- read.table("multivariate_statistics/data/data.txt", header = TRUE))
```

We start by finding distance matrixes among observation. In situation like this a common distance is jaccard: we compute it using `dist` function (euclidean distance done by default, so we specify "binary" as `method`)

[3]Starting from this idea people have tried to complicate the problem, by assuming we don't know the mixing proportion; number of cluster and extending to multiple variables: here we need to estimate mean vector and covariance matrices as well (not single mean and variances). It took a while but recently people invented methods to estimate EM algorithm.

```r
dist(data, method = "binary") # it computes lower triangular (symmetric)
```

```
##     1   2   3
## 2 0.6
## 3 0.6 0.0
## 4 1.0 1.0 1.0
```

2.5.2 Example 2: distance on mixed type of data

The dataset flower included in the cluster package contains 8 characteristics for 18 popular flowers:

- "`V1`" (factor) winters, is binary and indicates whether the plant may be left in the garden when it freezes;
- "`V2`" (factor) shadow, is binary and shows whether the plant needs to stand in the shadow;
- "`V3`" (factor) tubers, is asymmetric binary and distinguishes between plants with tubers and plants that grow in any other way;
- "`V4`" (factor) color, is nominal and specifies the flower's color (1 = white, 2 = yellow, 3 = pink, 4 = red, 5 = blue);
- "`V5`" (ordered) soil, is ordinal and indicates whether the plant grows in dry (1), normal (2), or wet (3) soil;
- "`V6`" (ordered) preference, is ordinal and gives someone's preference ranking going from 1 to 18;
- "`V7`" (numeric) height, is interval scaled, the plant's height in centimeters;
- "`V8`" (numeric) distance, is interval scaled, the distance in centimeters that should be left between the plants.

```r
# mixed-type data
library(cluster)
data(flower)
head(flower) # ?flower
```

```
##   V1 V2 V3 V4 V5 V6  V7 V8
## 1  0  1  1  4  3 15  25 15
## 2  1  0  0  2  1  3 150 50
## 3  0  1  0  3  3  1 150 50
## 4  0  0  1  4  2 16 125 50
## 5  0  1  0  5  2  2  20 15
## 6  0  1  0  4  3 12  50 40
```

Since variables are of mixed type, the dissimilarity matrix should be computed by using the Gower index. This computation can be done through the `daisy` function (`cluster` package), which requires, as input

- `x`: the data matrix of dimension $n \times p$;
- `metric`: the metric to be used. Possible options are "euclidean" (the default), "manhattan" and "gower";
- `type`: an optional list for specifying some (or all) of the types of the variables (columns) in x.

```
summary(d.flower <- daisy(flower, metric = 'gower',
                          type = list(asymm = 3))) # treat third variable as

## 153 dissimilarities, summarized :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1418  0.4164  0.5101  0.5098  0.6051  0.8875
## Metric :  mixed ;  Types = N, N, A, N, O, O, I, I
## Number of objects : 18

                                                    # asymmetric binary
```

we don't print `d.flower` for space but `summary.daisy` provides information on how the function considers the input features. In particular, here we have

- two `N`: nominal (factor),
- one A: asymmetric
- two O: ordinal (ordered factor),
- two I: interval scaled (numeric)

A double check on the actual nature of the variables is recommended.

2.5.3 Example 3: hierarchical clustering on quantitative variable

The dataset `Longley` included in the `AER` package consists of a time series on the number of people employed from 1947 to 1962. We want to look at employment

```
library(AER)
data(Longley)
head(longley <- as.data.frame(Longley))

##   employment price    gnp armedforces
## 1      60323  83.0 234289        1590
## 2      61122  88.5 259426        1456
## 3      60171  88.2 258054        1616
## 4      61187  89.5 284599        1650
## 5      63221  96.2 328975        3099
## 6      63639  98.1 346999        3594

## note how we prepare dataset to diminish code below
empl <- longley$employment
names(empl) <- as.character(1947:1962)
```

The aim of the analysis is to group observations (i.e. years) according to the `employment` level; it's quantitative, we use both the Euclidean and Manhattan distance to compute the distance (or dissimilarity) matrix.

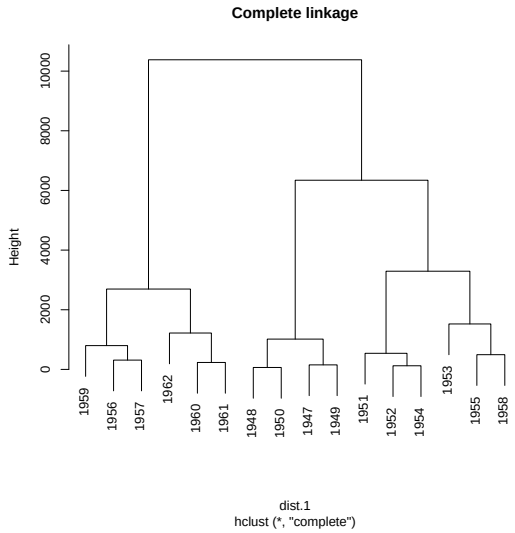
```
## now find the distance: different distance can be used (most used is
## euclidean)
## dist.1 <- dist(longley$employment, method = 'euclidean')
## dist.2 <- dist(longley$employment, method = 'manhattan')

dist.1 <- dist(empl, method = 'euclidean')
dist.2 <- dist(empl, method = 'manhattan')
```

Based on that, a hierarchical (agglomerative) clustering can be performed by using the hclust function. This function requires, as input:

- `d`: a dissimilarity matrix as produced by `dist`
- `method`: the agglomeration to be used. Common options are "`single`", "`complete`","`average`","`centroid`" and "`ward.D`".

2.5.3.1 Complete linkage clustering

```
(h <- hclust(dist.1)) # "complete" linkage (by default) on euclidian distance

##
## Call:
## hclust(d = dist.1)
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 16

plot(h, main = 'Complete linkage')
```



Values on the y-axis represent the distance between the nodes that are merged, which are contained in the `height` object, while the `merge` object contains all the single steps of clustering

```
h$merge ## how the tree was built during time?
```

```
##       [,1] [,2]
##  [1,]   -2   -4
##  [2,]   -6   -8
##  [3,]   -1   -3
##  [4,]  -14  -15
##  [5,]  -10  -11
##  [6,]   -9  -12
##  [7,]   -5    2
##  [8,]  -13    5
##  [9,]    1    3
## [10,]  -16    4
## [11,]   -7    6
## [12,]    8   10
## [13,]    7   11
## [14,]    9   13
## [15,]   12   14
```

The clustering steps are depicted from the bottom to the top of the plot.

- If in a row we have the negative sign in front of both numbers this means that the single observations are merged; for example, at step 1, units 2 (year 1948) and 4 (year 1950) are merged together.
- If, instead, in a row we have a negative value and a positive value this means that the observation with the minus sign has been merged to the sub-cluster created at the j-th step, where j is the number written in the second entry of the row. At step 7, for example, unit 5 is added to the cluster created at step 2 (i.e. that containing units 6 and 8).
- Finally, if in a row we have both values that are positive (e.g. p and j), this means that the sub clusters created in step p and j have been merged. For example, at step 9 the clusters created in steps 1 and 3 are merged together (i.e. the cluster containing units 2 and 4 and the one with units 1 and 3).

In order to find the most appropriate number of clusters, a common choice is to cut the tree by the largest difference between two nodes. As said `height` contains distance of cluster that are merged (y on dendogram). We can use it like this:

```
## height is important: look at distances between
h$height
```

```
##  [1]    65   122   152   233   312   494   540   798  1016  1220  1524  2694
## [13]  3292  6342 10380
```

```
(h.cl <- c(h$height[1], diff(h$height)))
```

```
##  [1]   65   57   30   81   79  182   46  258  218  204  304 1170  598 3050 4038
```

`h.cl` now contains differences of height between steps useful to cut the tree: we cut at the max distance

```
which.max(h.cl) # the largest difference was obtained in the last step
```

```
## [1] 15
```

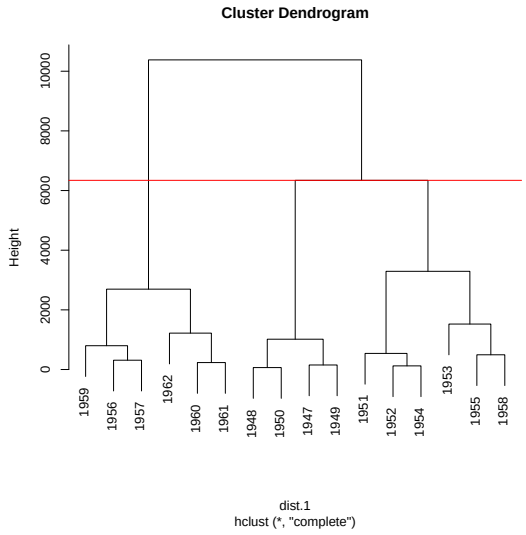
```
## extract the height to which it correspond the maximum distance
(h.max <- h$height[ which.max(h.cl) - 1]) # we want to cut the tree
```

```
## [1] 6342
```

```
## before the max distance so -1 is needed
```

The largest increase occurs at the last step of the merging process. This means that the best solution is the one with 2 clusters.

```
## final plot
plot(h)
abline(h = h.max, col = "red")
```




```
## with complete linkage the best number of group is two
```

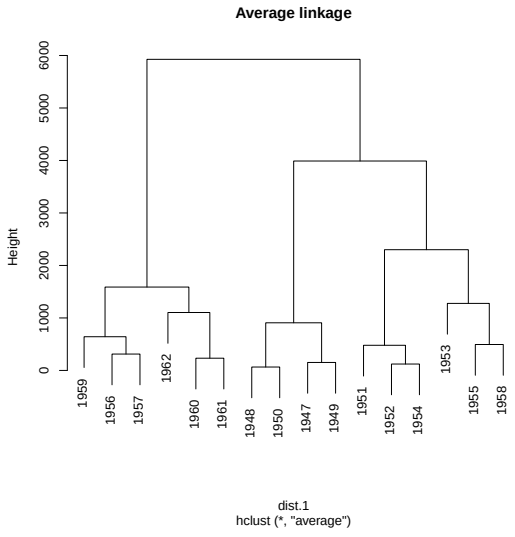
We can, thus, extract the classification/cut the tree so that we have two groups. In order to do that we can use the `cutree` function, specifying the desired number of groups or the cut height:

```
(cl <- cutree(tree = h, k = 2)) # cut to create two groups

## 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962
##    1    1    1    1    1    1    1    1    1    2    2    1    2    2    2    2

(cl <- cutree(tree = h, h = h.max)) # cut using an height: the h.max

## 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962
##    1    1    1    1    1    1    1    1    1    2    2    1    2    2    2    2
```

2.5.3.2 Average link method

Using methods other than complete linkage requires only to change methods at the beginning.

```
(h <- hclust(dist.1, method = "average"))

##
## Call:
## hclust(d = dist.1, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 16

plot(x = h, main="Average linkage")
```



In order to choose where to cut the tree, the differences in height values are evaluated

```
h.cl <- c(h$height[1], diff(h$height))
which.max(h.cl)
```

```
## [1] 15

h.max<-h$height[which.max(h.cl)-1]
```

Again, the number of clusters that seems most appropriate is 2, since the largest increase in the height values is at the last step of the merging process.

```
cl1 <-cutree(h, k = 2)
cl2 <- cutree(tree = h, h = h.max)
```

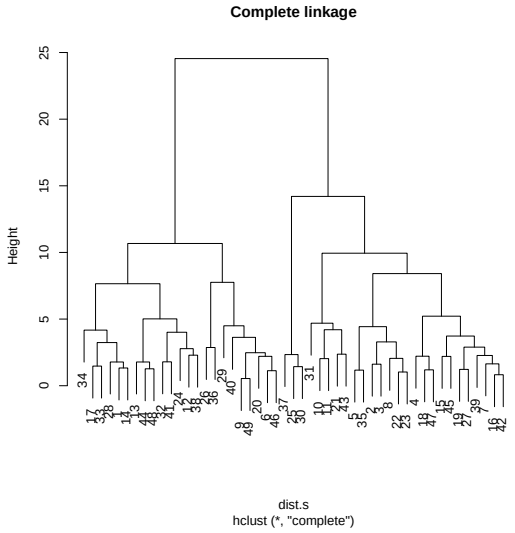
2.5.4 Example 4: sparrows dataset

The dataset `sparrows.dat` contains five external measurements from 49 sparrows that were found freezing during a storm. About half of the sparrows died, namely those in rows 22-49 of the data file. The aim of the analysis is to group sparrows on the basis of their external characteristics in order to establish whether sparrows who died tended to have more extreme measurements on some or all of the variables.

```
head(sparrow <- read.table("data/sparrows.dat", header = TRUE))

##   totL AlarE  bhL   hL   kL
## 1  156   245 31.6 18.5 20.5
## 2  154   240 30.4 17.9 19.6
## 3  153   240 31.0 18.4 20.6
## 4  153   236 30.9 17.7 20.2
## 5  155   243 31.5 18.6 20.3
## 6  163   247 32.0 19.0 20.9

dist.s <- dist(sparrow, method = 'euclidean')
```

2.5.4.1 Complete linkage

```
h <- hclust(d = dist.s, method = "complete")
plot(x = h, main = "Complete linkage")
```

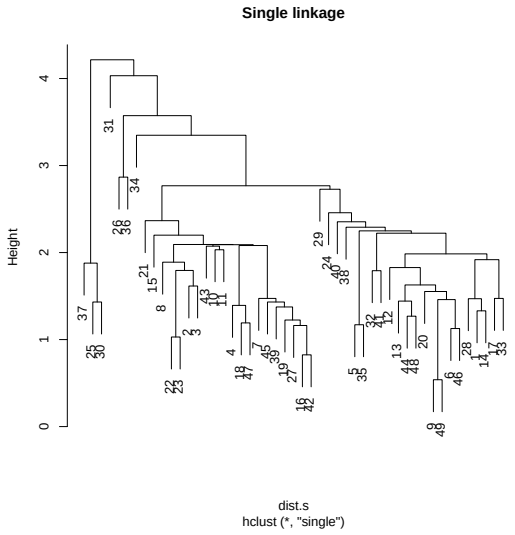



```
h.cl <- c(h$height[1], diff(h$height))
which.max(h.cl)

## [1] 48
```

According to the values of the height differences between nodes, the most appropriate number of clusters is 2, because the largest increase in terms of height is observed at the last step of the merging process.

2.5.4.2 Single linkage

```
h <- hclust(d = dist.s, method = "single")
plot(x = h, main = "Single linkage")
```



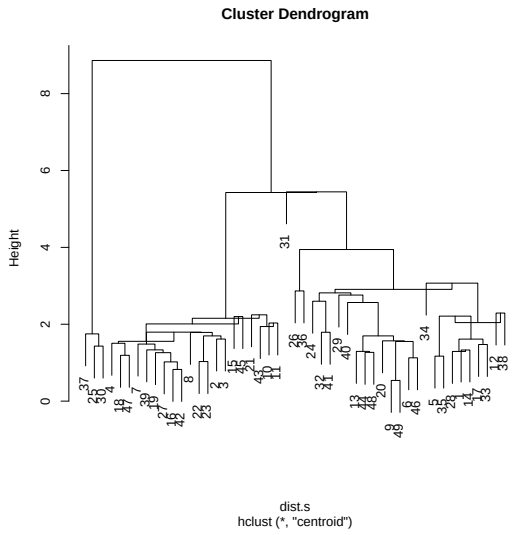
The single-linkage method produced asymmetric-looking clusters; this is the so called chaining effect, which refers to the tendency of the method to incorporate intermediate points between clusters into an existing cluster rather than initiating a new one. By its nature, this method is adapt only when clusters are well separated.

```
h.cl <- c(h$height[1], diff(h$height))
which.max(h.cl)

## [1] 1
```

According to the values of the height differences between nodes, the most appropriate number of clusters is 49, because at the first step there is the largest increase of the height value.

2.5.4.3 Centroid method

```
h <- hclust(dist.s, method="centroid")
plot(h)
```

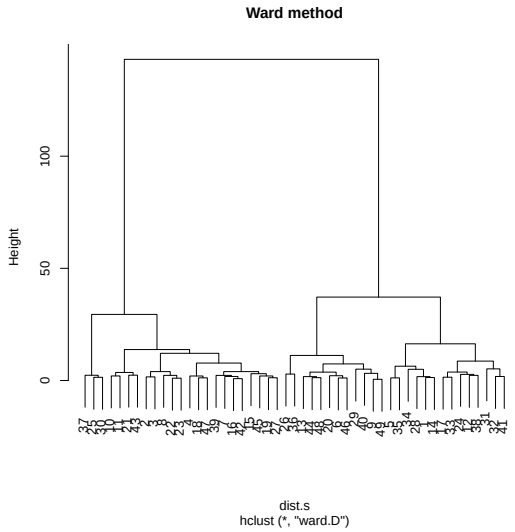



```
h.cl <- c(h$height[1], diff(h$height))
which.max(h.cl)

## [1] 48
```

According to the values of the height differences between nodes, the most appropriate number of clusters is 2, because the largest increase in terms of height is observed at the last step of the merging process.

2.5.4.4 Ward's method

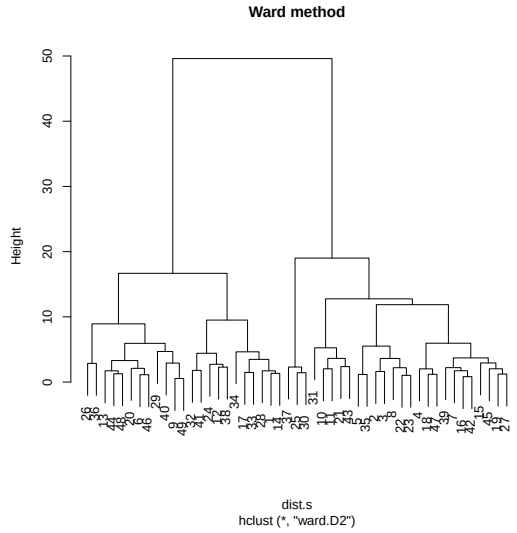
```
h <- hclust(d = dist.s, method = "ward.D")
plot(x = h, main = "Ward method")
```




```
h.cl <- c(h$height[1], diff(h$height))
which.max(h.cl)

## [1] 48
```

According to the values of the height differences between nodes, the most appropriate number of clusters is 2, because the largest increase in terms of height is observed at the last step of the merging process.
C'è anche un altro ward method btw

```
h <- hclust(d = dist.s, method = "ward.D2")
plot(x = h, main = "Ward method")
```




```
h.cl <- c(h$height[1], diff(h$height))
which.max(h.cl)

## [1] 48
```

2.6 K-means

Now try to use the k-means cluster analysis to check whether there are two clusters, one containing the sparrows which survived and the other the sparrows which died:

```
set.seed(1234) ## if we run this function we could get different results, set the see
km <- kmeans(x = sparrow, centers = 2) # x is the data, we start with two groups
## extract the cluster k-means built
km$cluster # info on clusters: there's not a link between clustering and the

##  [1] 1 2 2 2 2 1 2 2 1 2 2 1 1 1 2 2 1 2 2 1 2 2 2 1 2 1 2 1 1 2 1 1 1 1 2 1 2 1
## [39] 2 1 1 2 2 1 2 1 2 1 1

# actual dead: (dead are the last one)
km$centers # info on centroids

##       totL    AlarE      bhL       hL       kL
## 1 160.9167 245.4583 31.90417 18.80833 21.29583
## 2 155.1600 237.3600 31.03200 18.14400 20.37600
```

The clustering vector seems not to reflect the actual partition of dead/alive sparrows.

The k-means algorithm is then run by allowing the number of clusters to vary from 2 to 5. The goodness of each run can be measured in terms of Average silhouette width and Pearson-Gamma index. These two measures (that should

be maximized) can be obtained with the function `cluster.stats` in package `fpc`.

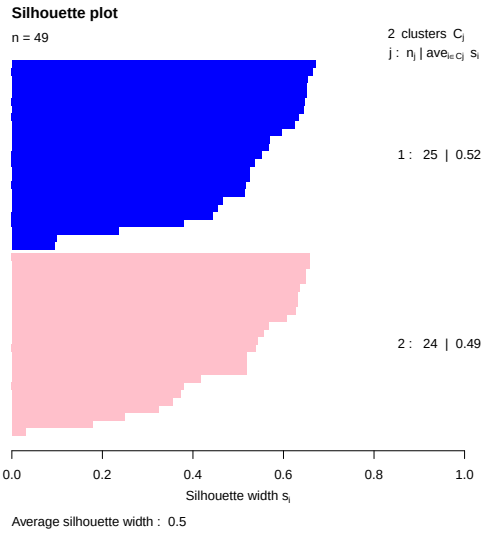
```
## install.packages("fpc")
library(fpc)

## average silhouette and pearson gamma for several n of k groups build a
## matrix of

indexes <- matrix(NA, nrow = 5, ncol = 2)
for (k in 2:5){
  set.seed(1234)
  km <- kmeans(x = sparrow, centers = k)
  stats <- cluster.stats(dist.s, # distance as first object, clustering as second
                         km$cluster)
  indexes[k, ] <- c(
    stats$avg.silwidth, ## average silhouette width in first col
    stats$pearsongamma   ## average pearson gamma second col
  )
}

colnames(indexes) <- c("AVG", "PG")
rownames(indexes) <- c("cl = 1", "cl = 2", "cl = 3", "cl = 4", "cl = 5")
indexes

##               AVG        PG
## cl = 1         NA        NA
## cl = 2 0.5046398 0.6496322
## cl = 3 0.3375809 0.5476798
## cl = 4 0.3925813 0.6049124
## cl = 5 0.3952145 0.5493019
```

We need to look by column: maximum value for both indicator is k=2 (at least with this seed); both indexes agree. So according to the values of the two indexes, the k-means with two clusters produces more homogeneous groups.

A meaningful representation of the clustering outcome is the so called silhouette plot. On the x-axis it shows the silhouette width (i.e. a measure of how much similar an observation is to the others in the same cluster) for each observation in the corresponding cluster; units in the same cluster are plotted in decreasing order according to their silhouette value. Different clusters are separately plotted. Furthermore, it reports the number of observations in each cluster and the average silhouette width of the classification. In order to produce a silhouette plot we need to use function `silhouette` (`cluster` package). It requires, as first argument `x`, the classification vector produced by the clustering algorithm; the dissimilarity/distance matrix should be provided as second argument `dist`.

```
## plot silhouette of the two groups
library(cluster)
dist.s <- dist(sparrow, method = "euclidean")
```

```
sil <- silhouette(kmeans(x = sparrow, centers = 2)$cluster, dist.s)
plot(sil, col = c("blue", "pink"), main ="Silhouette plot")
```



2.6.1 Life expectancy data

Dataset lifeexp.dat contains life expectancy in the 1960s distinguished by country, age and sex. In this case, the aim of the analysis is to group countries according to their level of life expectancy in the 1960s.

```
head(lifeexp <- read.table("data/lifeexp.dat", header = TRUE))
```

```
##      Country LifeExp Television Physician FemaleLife MaleLife
## 1  Argentina    70.5        4.0       370         74       67
## 2 Bangladesh    53.5      315.0      6166         53       54
## 3     Brazil    65.0        4.0       684         68       62
## 4     Canada    76.5        1.7       449         80       73
## 5      China    70.0        8.0       643         72       68
## 6   Colombia    71.0        5.6      1551         74       68
```

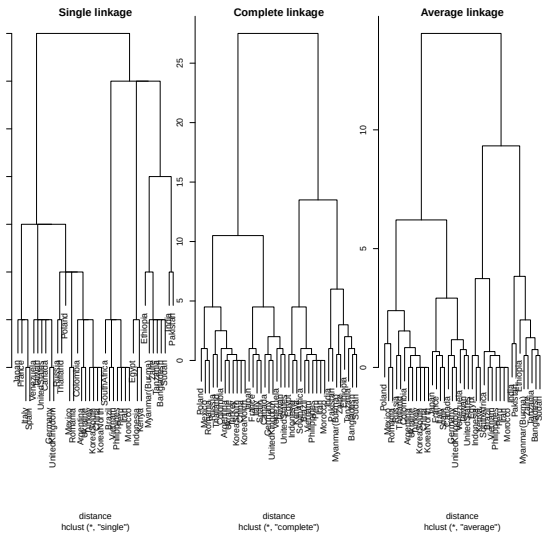
```
life <- lifeexp$LifeExp
names(life) <- lifeexp$Country
life
```

```
##      Argentina     Bangladesh         Brazil         Canada          China
##           70.5           53.5           65.0           76.5           70.0
##       Colombia          Egypt       Ethiopia         France        Germany
##           71.0           60.5           51.5           78.0           76.0
##          India      Indonesia           Iran          Italy          Japan
##           57.5           61.0           64.5           78.5           79.0
##          Kenya     KoreaNorth     KoreaSouth         Mexico        Morocco
##           61.0           70.0           70.0           72.0           64.5
## Myanmar(Burma)       Pakistan           Peru    Philippines         Poland
```

```
##          54.5          56.5          64.5          64.5          73.0
##       Romania        Russia  SouthAfrica         Spain         Sudan
##          72.0          69.0          64.0          78.5          53.0
##        Taiwan      Tanzania      Thailand        Turkey       Ukraine
##          75.0          52.5          68.5          70.0          70.5
##  UnitedKingdom UnitedStates     Venezuela       Vietnam         Zaire
##          76.0          75.5          74.5          65.0          54.0
```

Hierarchical clustering

```
## calculate distance
distance <- dist(life, method = "euclidean")

## dendogram
par(mfrow=c(1,3), mai = c(1, 0.1, 0.1, 0.1))
h <- hclust(d = distance, method = "single") # single linkage
plot(x = h, main = "Single linkage")
h <- hclust(d = distance, method = "complete") # complete linkage
plot(x = h, main = "Complete linkage")
h <- hclust(d = distance, method = "average") # average linkage
plot(x = h, main ="Average linkage")
```



It is important to underline the different results produced by the three methods:

- the singe linkage method shows a high degree of asymmetry - an example of 'chaining', i.e. result of 'prematurely' combining individuals/clusters using the minimum distance criteria which defines single linkage;
- the complete linkage method is much more balanced in the way it forms clusters, producing four or five clear spherical and compact clusters;

- the average linkage method returns clusters which again show asymmetry, but not as pronounced as single linkage.

Suppose to cut the complete linkage dendrogram at height 8: it would yield 4 clusters. Try to construct four clusters using the k-means algorithm as follows

2.6.1.1 k-means

```
set.seed(1234)
clusters.km <- kmeans(life, centers = 4)
clusters.km$cluster
```

```
##      Argentina     Bangladesh         Brazil         Canada          China
##              4              3              1              4              4
##       Colombia          Egypt       Ethiopia         France        Germany
##              4              2              3              4              4
##          India      Indonesia           Iran          Italy          Japan
##              2              2              1              4              4
##          Kenya     KoreaNorth     KoreaSouth         Mexico        Morocco
##              2              4              4              4              1
## Myanmar(Burma)       Pakistan           Peru    Philippines         Poland
##              3              3              1              1              4
##        Romania         Russia    SouthAfrica          Spain          Sudan
##              4              1              1              4              3
##         Taiwan       Tanzania       Thailand         Turkey        Ukraine
##              4              3              1              4              4
##  UnitedKingdom   UnitedStates      Venezuela        Vietnam          Zaire
##              4              4              4              1              3
```

```
group <- list()
for(i in 1:4) group[[i]] <- countries[clusters.km$cluster == i]
```

```
## Error:  oggetto 'countries' non trovato
```

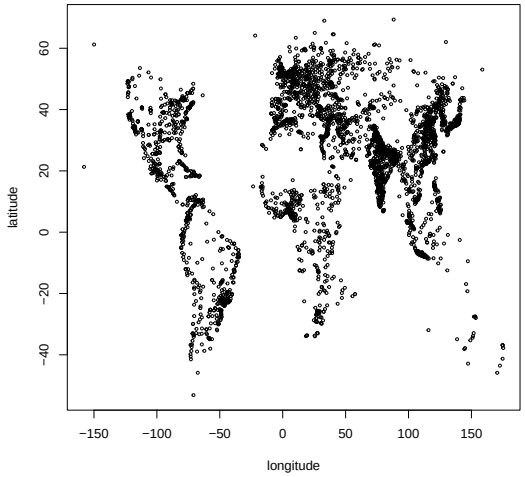
```
group
```

```
## list()
```

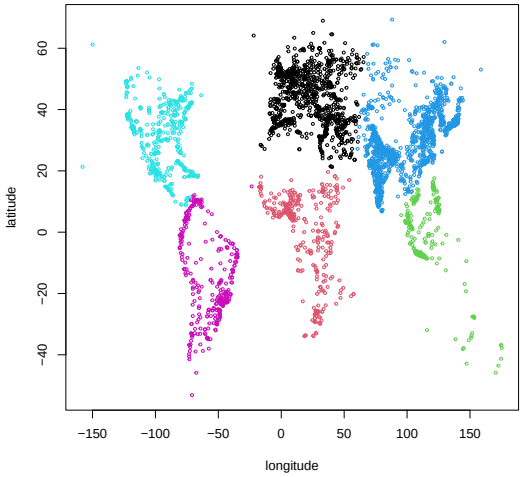
From our limited knowledge of demographic properties, the groupings appear consistent, eg. developing countries are grouped together, developed countries are grouped together.

2.6.2 Example 6 - City coordinates data

```
library(mdsr)
data(world_cities)
big_cities <- world_cities[order(world_cities$population,decreasing = TRUE)[1:4000],
                           c("longitude", "latitude")]
plot(big_cities, cex = 0.5)
```




```
set.seed(15)
kmcities <- kmeans(x = big_cities, centers = 6)
str(kmcities)

## List of 9
##  $ cluster     : int [1:4000] 4 1 6 4 5 4 4 4 4 4 ...
##  $ centers     : num [1:6, 1:2] 22.4 18.9 115.9 102.1 -94.5 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:6] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:2] "longitude" "latitude"
##  $ totss       : num 24082261
##  $ withinss    : num [1:6] 487843 171571 112232 838329 203611 ...
##  $ tot.withinss: num 1963878
##  $ betweenss   : num 22118383
##  $ size        : int [1:6] 1106 355 296 1297 554 392
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

plot(big_cities, col = kmcities$cluster, cex = 0.5)
```



Choosing the number of groups according to Average Silhouette Width and Pearson Gamma coefficients

```
library(fpc)
indexes <- matrix(NA, 10, 2)
for (cl in 2:10){
  set.seed(15)
  clustering <- kmeans(big_cities, centers = cl)
  stats <- cluster.stats(dist(big_cities), clustering$cluster)
  indexes[cl,] <- c(stats$avg.silwidth, stats$pearsongamma)
}
colnames(indexes) <- c("AVG", "PG")
rownames(indexes) <- c("cl=1","cl = 2", "cl = 3", "cl = 4", "cl = 5","cl = 6",
                       "cl = 7", "cl = 8","cl = 9","cl = 10")
indexes
```

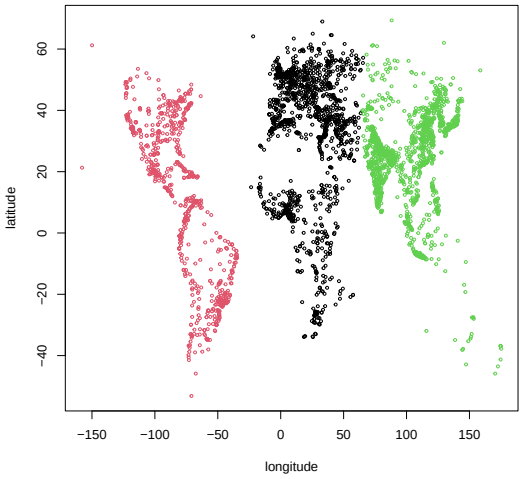
```
##                AVG        PG
## cl=1            NA        NA
## cl = 2   0.5215506 0.6430393
## cl = 3   0.5792505 0.6915144
## cl = 4   0.4550244 0.6301318
## cl = 5   0.4412401 0.6174299
## cl = 6   0.4293478 0.5814931
## cl = 7   0.5217120 0.5273805
## cl = 8   0.5012090 0.5137080
## cl = 9   0.5041452 0.4708019
## cl = 10  0.5031857 0.4666986
```

```
which.max(indexes[,1])
```

```
## cl = 3
##      3
```

```
which.max(indexes[,2])
```

```
## cl = 3
##      3

set.seed(15)
km_c <- kmeans(big_cities, centers=3)
plot(big_cities, col = km_c$cluster, cex = 0.5)
```



Capitolo 3

Principal component analysis

With qualitative data correspondence analysis to do the same shit

3.1 Linear combinations

Definition 3.1.1 (Linear combination)**.** A linear combination is sum of vector with weights given from another vector. Assuming we have a random vector X including p random variables (dimension $p \times 1$) a and a vector of constant terms:

$$\underset{p\times 1}{\mathbf{x}} = \begin{bmatrix} X_1 \\ \dots \\ X_i \\ \dots \\ X_p \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_1 \\ \dots \\ a_i \\ \dots \\ a_p \end{bmatrix} \in \mathbb{R}^p$$

a linear combination Y of the two is

$$Y = \mathbf{a}^\top \mathbf{x} = a_1 X_1 + \ldots + a_i X_i + \ldots a_p X_p$$

We have that Y is a linear combination of the random variables in $\mathbf{x}$ with weights given by the components/coefficients of $\mathbf{a}$; Y is a scalar random variable (has dimension 1×1).

Important remark 19. We assume that:

- our random vector $\mathbf{x}$ has expected value/means $\mathbb{E}[x] = \boldsymbol{\mu}$: each X_i has its own expected values and we collect all these expected value in a vector

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \dots \\ \mu_i \\ \dots \\ \mu_p \end{bmatrix} = \begin{bmatrix} \mathbb{E}[X_1] \\ \dots \\ \mathbb{E}[X_i] \\ \dots \\ \mathbb{E}[X_p] \end{bmatrix}$$

- our vector $\mathbf{x}$ has covariance matrix denoted by $\boldsymbol{\Sigma}$ which is a $p \times p$ matrix with variances on the diagonal and covariances out of the diagonal. Since it's square and symmetric for simplicity people just report the upper

matrix:

$$\mathrm{Var}[X] = \underset{p\times p}{\boldsymbol{\Sigma}} = \begin{bmatrix} \sigma_1^2 & \dots & \sigma_{1i} & \dots & \sigma_{1p} \\ & & \sigma_i^2 & \dots & \sigma_{ip} \\ & & & & \sigma_p^2 \end{bmatrix}$$

3.1.1 Variance of linear combinations

Important remark 20. What happens to expected value and variance/covariance if we take a linear combination of the variable:

- for the expected value we have

$$\mathbb{E}[Y] = \mathbb{E}\left[\mathbf{a}^\top \mathbf{x}\right] = a^\top \mathbb{E}[\mathbf{x}] = \mathbf{a}^\top \boldsymbol{\mu} = a_1\mu_1 + \ldots + a_i\mu_i + \ldots + a_p\mu_p$$

  so the expected value of a linear combination is the linear combination of the expected values (and is a scalar)

- regarding the variance, to obtain the analog of the square we have to multiply pre and post for $\mathbf{a}$ respecting dimensions

$$\mathrm{Var}[Y] = \mathrm{Var}\left[\mathbf{a}^\top \mathbf{x}\right] \overset{(1)}{=} \underset{1\times p}{\mathbf{a}^\top} \underset{p\times p}{\mathrm{Var}[\mathbf{x}]} \underset{p\times 1}{\mathbf{a}} = \underbrace{\mathbf{a}^\top \boldsymbol{\Sigma} \mathbf{a}}_{1\times 1}$$

  where in (1) we used the equivalent of $\mathrm{Var}[aX] = a^2 \mathrm{Var}[X]$ in case of random vector instead of single variable. So the variance is a scalar as well

Considering the simple case of $\mathbf{x}$ composed by two variable $\mathbf{x} = [X_1, X_2]^\top$, this means that

$$\begin{aligned} \mathrm{Var}[Y] = \mathbf{a}^\top \boldsymbol{\Sigma} \mathbf{a} &= \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \\ &= \begin{bmatrix} a_1\sigma_1^2 + a_2\sigma_{12} & a_1\sigma_{12} + a_2\sigma_2^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \\ &= a_1^2 + \sigma_1^2 + a_1a_2\sigma_{12} + a_1a_2\sigma_{12} + a_2^2\sigma_2^2 \\ &= a_1^2\sigma_1^2 + 2a_1a_2\sigma_{12} + a_2^2\sigma_2^2 \end{aligned}$$

Focusing

- on the first element a_1 thinking variance as function of a_1 what happens to the variance when a_1 changes? This is an upward parabola (coefficient σ_1^2 is necessarily positive being a variance) : as a_1 varies the variance describe an upward parabola

- focusing on a_2 it's the same thing, again it's an upward parabola

In our bidimensional space, variance as fuction of a_1, a_2 (taken as x, y) is a cup: the variance goes to infinity, does not have a finite maximum. We can increase the variance of linear combination just by simply increasing the value of a_1 and a_2

In other words, in the multidimensional space the variance of a linear combination can be increased just by using a vector $\mathbf{a}$ with larger norm: the larger the norm of the vector $\mathbf{a} = (a_1, a_2)$ the larger the variance of the linear combination it produces.
So the variance of a linear combination does not have a finite maximum. Keeping the data as fixed we can increase the variance of a linear combination by simply increasing the norm of the vector a containing coefficient of the linear combination $\mathbf{a}$
This is something we don't like: we want something that is linked to the data for an increase in variance. This is a point important for principal component.

Important remark 21. To take the linear algebra approach the variance $\operatorname{Var}[Y] = \mathbf{a}^\top \Sigma \mathbf{a}$ is a quadratic form, represented by a positive semidefinite matrix.
So a positive semidefinite quadratic form does not have a finite maximum.

Remark 35. These are two different way of saying the same and are the preliminaries for PCA.

3.1.2 Multiple linear combinations

Remark 36. We have more than 1 linear combinations: in that case we can collect the weights in a matrix called $\boldsymbol{A}$, that will be a $p \times q$ where q is the number of linear combinations we're interested in.[1]

Important remark 22. A vector of q linear combinations $\mathbf{y}$ $(q \times 1)$ can be obtained as

$$\underset{q\times 1}{\mathbf{y}} = \underset{q\times p}{\mathbf{A}^\top}\ \underset{p\times 1}{\mathbf{x}}$$

Properties of mean and variance of transformations applies again so:

$$\mathbb{E}[Y] = \mathbf{A}^\top \mathbb{E}[X] = \mathbf{A}^\top \boldsymbol{\mu}$$
$$\operatorname{Var}[Y] = \mathbf{A}^\top \operatorname{Var}[X]\, \mathbf{A} = \mathbf{A}^\top \boldsymbol{\Sigma} \mathbf{A}$$

with $\mathbb{E}[Y]$ a $q \times 1$ vector and $\operatorname{Var}[Y]$ no longer a scalar but a $q \times q$ matrix (Since we have more than one random variable there is not only the variances but also the covariances).

Example 3.1.1. Assume we have a bivariate random vector $\mathbf{x} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ with mean $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ and variance/covariance matrix $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$.
Now let's consider the following linear combinations

$$Y_1 = X_1 - X_2$$
$$Y_2 = X_1 + X_2$$

Derive the expected values and the covariance matrix of $Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$

[1] Warning: we've no enough letters so the notation is consistent within each topic. This matrix $\mathbf{A}$ has to do nothing with the centering matrix seen before. In this chapter $\mathbf{A}$ is the matrix of coefficients

Example 3.1.2. Consider three independent standardized variables Z_1, Z_2, Z_3. Assume we transform them as

$$\begin{aligned} Y_1 &= Z_1 \\ Y_2 &= Y_1 + 0.01 \cdot Z_2 \\ Y_3 &= 10 \cdot Z_3 \end{aligned}$$

derive the covariance matrix of $Y = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$

3.2 PCA

Principal component analysis has been invented by Pearson in 1901, but the way we'll see it is not based on the development of Hotelling in 1933. They wanted to solve two different problem but found to obtain the same solutions, they've both invented it:

- Pearson wanted to solve a regression problem (1901 "On lines and planes of closest fit to a system of points"); he wanted to find a line that best fit the point, but in a special condition where he had errors in the predictors[2].
- Hotelling OTOH have too many variables and can't find a way to manage them so wanted to summarize them producing a smaller number of variables that keep as much info as in the original sets. Hotelling found a low dimensional function (linear combination) of the observed data that preserves as much of the variability as possible. The approach is the same used at today.

They didn't have machines to do calculations, but used cleverly linear algebra, eigenvalue and eigenvector to tackle the problem.

The basic idea of principal component analysis is to find a small number of linear combinations of the observed variables which explain most of the variation in the data. The first principal component is the linear combination with maximal variance; we are essentially searching for a dimension along which the observations are maximally separated or spread out. The second principal component is the linear combination with maximal variance in a direction orthogonal to the first principal component, and so on.

3.2.1 First principal component

Let's consider a p-dimensional random vector $\mathbf{x}$, with expected value $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, and define a single linear combination of the variables (we see other combination afterward) such as

$$y_1 = \mathbf{a}_1^\top \mathbf{x}$$

The point is that we want to find a linear combination (so determines the values of a_1), such that the variance of the combination $\mathrm{Var}[y_1] = \mathbf{a}_1^\top \boldsymbol{\Sigma} \mathbf{a}_1$ is maximum.

[2] While in regression models one assumes that the x's are fixed without errors, Pearson was in a different situation having error in x

Problem is that, as we've seen before, a maximum on variance does not exists: it is enough to take $\mathbf{a}_1$ with norm higher and higher.
What interests us is actually the direction of the vector $\mathbf{a}_1$: in that case I can put a constraint on the norm of $\mathbf{a}_1$ (in other words to have a fair comparison between vectors to find the higher variance, we put a constraint on the norm). In this case a unit norm vector is enough to identify a direction in space, so the constraint that we put is that $\|\mathbf{a}_1\| = 1$, that is $\mathbf{a}_1^\top \mathbf{a}_1 = 1$.
So finally the problem we need to solve is a *constrainted optimization problem*. That is we want to choose $\mathbf{a}_1$ to maximize $\text{Var}[y_1] = \mathbf{a}_1^\top \mathbf{\Sigma} \mathbf{a}_1$ under the constraint $\|\mathbf{a}\| = \mathbf{a}_1^\top \mathbf{a}_1 = 1$.
This can be solved by using Lagrange multiplier; the problem above is equivalent to maximize this function:

$$\phi = \mathbf{a}_1^\top \mathbf{\Sigma} \mathbf{a}_1 - \lambda_1 (\mathbf{a}_1^\top \mathbf{a}_1 - 1)$$

and looking for the vector a_1 that maximizes it (λ_1 is a Lagrange multiplier). This optimization problem can be solved by differentiating ϕ with respect to $\mathbf{a}_1$ and equating to 0 all the partial derivatives: in other words $\mathbf{a}_1$ is our unknown; we check/consider all possible unit norm vector, and put a system of p equations (we need to take p different derivatives, one for each element of $\mathbf{a}_1$). So:

$$\frac{\partial \phi}{\partial \mathbf{a}_1} = 2\mathbf{\Sigma} \mathbf{a}_1 - 2\lambda_1 \mathbf{a}_1$$

which equaled to $\mathbf{0}$ yields

$$\mathbf{\Sigma} \mathbf{a}_1 = \lambda_1 \mathbf{a}_1$$

That is we need to find the $\mathbf{a}_1$ that satisfy the equation above. One can easily recognize in this identity the relationship between the eigenvalues and the eigenvectors of the covariance matrix $\mathbf{\Sigma}$: λ_1 is an eigenvalue of $\mathbf{\Sigma}$ and $\mathbf{a}_1$ is the corresponding eigenvector.
So the equality above holds if $\mathbf{a}_1$ is eigenvector of $\mathbf{\Sigma}$ and λ_1 is the corresponding eigenvalue and thus *the solution to the problem consist in finding a pair of eigenvalue-eigenvector* for the matrix $\mathbf{\Sigma}$.
However, how many eigenvalues has $\mathbf{\Sigma}$? Since it is $p \times p$ it is expected to have p eigenvalues and corresponding eigenvectors. But we are interested in the just one: which pair of eigenvalues-vector to choose/solves my problem/to be used for the first component?
In order to answer let's premultiply both sides by $\mathbf{a}_1^\top$

$$\begin{aligned} \mathbf{\Sigma} \mathbf{a}_1 &= \lambda_1 \mathbf{a}_1 \\ \mathbf{a}_1^\top \Sigma \mathbf{a}_1 &= \lambda_1 \underbrace{\mathbf{a}_1^\top \mathbf{a}_1}_{=1} \\ \mathbf{a}_1^\top \Sigma \mathbf{a}_1 &= \lambda_1 \end{aligned}$$

Some points:

- λ_1 coincides with the variance of y_1 i.e. with the quantity we want to maximize

- therefore in order to derive the linear combination having the largest variance, we simply need to consider the largest eigenvalue of $\mathbf{\Sigma}$ and $\mathbf{a}_1$ will be the corresponding eigenvector
- this if we had to choose just one, we'll see the remainings later

Definition 3.2.1 (First principal component)**.** The linear combination of having $\mathbf{a}_1$ as vector of coefficients is called *first principal component*:

$$y = \mathbf{a}_1\mathbf{x}$$

Remark 37. In other words we can say that the first principal component is the linear combination of the observed variables *having the largest variance*. It's variance will be the largest eigenvalue of $\mathbf{\Sigma}$ and the optimal vector of coefficients $\mathbf{a}_1$ will be the corresponding eigenvector.

Remark 38 (Recap). Main point so far:

- principal Components are linear combinations of observed variable: main tool we use is linear combinations.
- any linear combination does not have finite variance; the variance is positive semidefinite ($\mathbf{\Sigma}$) and has no finite maximum. Variance is an upward parabola.
- maximizing without constraint is therefore risky, we could increase fictitiously variance without reference to the data explained. What we're interested in is not the norm of the vector but the direction of the vector; the unit norm is enough to identify direction in space and to make fair comparison we compare vector having the same norm, for simplicity we take unit norm.
- we want to maximize the variance under the constraint looking only at unit norm vectors of combination. We use Lagrange multiplier to do the maximization.
- to find the maximum we take first derivative equal to 0: our function has a maximum because we constraint on a unit norm vector the maximum is reached with eigenvalue/eigenvector equality
- problem is to find suitable pair of eigenvalue eigenvector: which one to consider? the higher, since it is the variance. So we've identified the first principal component

3.2.2 Following components

We've spoken of the first principal components: maybe it's very improbable that a single linear combination is enough (to explain total variance of starting variable) and we need more than one.
As said $\mathbf{\Sigma}$ has p pairs eigenvalue-eigenvector: with large number of variables it is difficult that a single combination is enough to save all the variability of the higher dimensional space. So we're interested in looking at a second linear combination.

How can we find the second principal component? It will still be a linear combination:

$$y_2 = \mathbf{a}_2^\top \mathbf{x}$$

with variance

$$\text{Var}\,[y_2] = \mathbf{a}_2^\top \mathbf{\Sigma} \mathbf{a}_2$$

but:

- we want the PCs to be orthogonal/uncorrelated;
- so we need to constraint the optimization search in the orthogonal space to what found at the first step

The Lagrange function will be constructed:

- to maximize $\mathbf{a}_2^\top \Sigma \mathbf{a}_2$
- under the unit norm constraint $\mathbf{a}_2^\top \mathbf{a}_2 = 1$ *and*
- under the constraint that $\mathbf{a}_2$ is orthogonal to $\mathbf{a}_1$, that is $\mathbf{a}_1^\top \mathbf{a}_2 = \mathbf{a}_2^\top \mathbf{a}_1 = 0$.

So we obtain:

$$\phi = \mathbf{a}_2^\top \Sigma \mathbf{a}_2 - \lambda_2(\mathbf{a}_2^\top \mathbf{a}_2 - 1) - \lambda_3(\mathbf{a}_1^\top \mathbf{a}_2)$$

above:

- the first constraint to have unit norm in $\mathbf{a}_2$;
- the second is to impose that $\mathbf{a}_1^\top \mathbf{a}_2 = 0$ (we removed -0) so that $\mathbf{a}_1$ and $\mathbf{a}_2$ are orthogonal.

In order to find $\mathbf{a}_2$ we can derive ϕ respect to $\mathbf{a}_2$ which is our unknown

$$\frac{\partial \phi}{\partial \mathbf{a}_2} = 2\mathbf{\Sigma}\mathbf{a}_2 - 2\lambda_2 \mathbf{a}_2 - \lambda_3 \mathbf{a}_1 \tag{3.1}$$

then we set our derivative equal to $\mathbf{0}$. This problem can be simplified a little bit. Let's premultiply both sides by $\mathbf{a}_1^\top$, we have that

$$2\mathbf{a}_1^\top \mathbf{\Sigma}\mathbf{a}_2 - 2\lambda_2 \mathbf{a}_1^\top \mathbf{a}_2 - \lambda_3 \underbrace{\mathbf{a}_1^\top \mathbf{a}_1}_{=1} = \mathbf{0} \tag{3.2}$$

Now considering $\mathbf{a}_1^\top \mathbf{\Sigma}$, remembering that $\mathbf{a}_1$ is an eigenvector of $\mathbf{\Sigma}$ (so $\mathbf{\Sigma}\mathbf{a}_1 = \lambda_1 \mathbf{a}_1$ and thus by transposing $\mathbf{a}_1^\top \mathbf{\Sigma} = \lambda_1 \mathbf{a}_1^\top$) we have

$$\mathbf{a}_1^\top \mathbf{\Sigma} = (\mathbf{\Sigma}\mathbf{a}_1)^\top = \lambda_1 \mathbf{a}_1^\top$$

Back to 3.2 we can simplify as

$$2\lambda_1 \underbrace{\mathbf{a}_1^\top \mathbf{a}_2}_{=0} - 2\lambda_2 \underbrace{\mathbf{a}_1^\top \mathbf{a}_2}_{=0} - \lambda_3 \underbrace{\mathbf{a}_1^\top \mathbf{a}_1}_{=1} = 0$$
$$-\lambda_3 = 0$$

This implies that $\lambda_3 = 0$. And back to 3.1 this means that therefore the derivative of ϕ with respect to $\mathbf{a}_2$ will be

$$\frac{\partial \phi}{\partial \mathbf{a}_2} = 2\boldsymbol{\Sigma}\mathbf{a}_2 - 2\lambda_2\mathbf{a}_2 = \mathbf{0}$$
$$\boldsymbol{\Sigma}\mathbf{a}_2 = \lambda_2\mathbf{a}_2$$

and this is again an eigenvalue-eigenvector relationship. If we premultiply by $\mathbf{a}_2^\top$

$$\underbrace{\mathbf{a}_2^\top \Sigma \mathbf{a}_2}_{\mathrm{Var}[Y_2]} = \lambda_2 \underbrace{\mathbf{a}_2^\top \mathbf{a}_2}_{=1}$$

so

$$\mathrm{Var}\left[Y_2\right] = \lambda_2$$

as we want the linear combination of $\mathbf{x}$ having the largest variance in the orthogonal complement of $\mathbf{a}_1$ we take λ_2 as the second largest eigenvalue and $\mathbf{a}_2$ will be the corresponding eigenvector; thus the second PC will be the linear combination of $\mathbf{x}$ with $\mathbf{a}_2$.

Remark 39. The process above can be continued for all principal components. Going on this way, for p variables we can find up to p principal components.
In general, the k-th PC of $\mathbf{x}$ is $y_k = \mathbf{a}_k^\top \mathbf{x}$ and $\mathrm{Var}\left[\mathbf{a}_k^\top \mathbf{x}\right] = \lambda_k$ where λ_k is the k-th largest eigenvalue of $\boldsymbol{\Sigma}$, and $\mathbf{a}_k$ is the corresponding eigenvector.

Remark 40. In summary: starting from the p-dimensional random vector $\mathbf{x}$ we have obtained a new p-dimensional random vector $\mathbf{y}$, such that $\mathbf{y} = \mathbf{A}^\top \mathbf{x}$ where $\mathbf{A}$ is an orthonormal matrix whose k-th column is the eigenvector of $\boldsymbol{\Sigma}$ corresponding to the k-th largest eigenvalue. Thus, the PCs are defined by an *orthonormal linear transformation* of $\mathbf{x}$.
In this section we explode this.

3.3 Obtaining PCs in practice

In practice how to obtain principal components? For the first one we want to find the couple $\lambda_1, \mathbf{a}_1$ which is solution to

$$\boldsymbol{\Sigma}\mathbf{a}_1 = \lambda_1\mathbf{a}_1$$
$$\boldsymbol{\Sigma}\mathbf{a}_1 - \lambda_1\mathbf{a}_1 = \mathbf{0}$$

now we collect $\mathbf{a}_1$ obtaining the identity matrix of the same size

$$(\boldsymbol{\Sigma} - \lambda_1\mathbf{I})\mathbf{a}_1 = \mathbf{0}$$

It's a linear equation system which is homogeneous (constant term is 0): it admits a nontrivial solutions (eg $\mathbf{a}_1 = \mathbf{0}$) if and only if the matrix of coefficient is singular, that is if its determinant is zero

$$\det\left(\boldsymbol{\Sigma} - \lambda_1\mathbf{I}\right) = 0$$

This means that λ_1 should be *a root of the characteristic polinomial* (an eigenvalue) and $\mathbf{a}_1$ will be the corresponding eigenvector. So finding the principal components amounts to solve eigenvalue/vector problem or to solve a linear equation system.

Example 3.3.1. Considering a bivariate vector

$$\mathbf{x} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad \mathrm{Cov}\,(\mathbf{x}) = \mathbf{\Sigma} = \begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix}$$

We want to find the principal components. What we need to solve is

$$\begin{aligned} (\mathbf{\Sigma} - \lambda_1 \mathbf{I})\mathbf{a}_1 &= \mathbf{0} \\ \left[\begin{bmatrix} 5 & 2 \\ 2 & 2 \end{bmatrix} - \lambda_1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right] \mathbf{a}_1 &= \mathbf{0} \\ \begin{bmatrix} 5-\lambda_1 & 2 \\ 2 & 2-\lambda_1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

where λ_1 is our unknown and we rewrote $\mathbf{a}_1$, the vector of coefficients we're looking for, as $\mathbf{a}_1 = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$.
In order to have a nontrivial solution we need that determinant to be null

$$\begin{aligned} \begin{vmatrix} 5-\lambda_1 & 2 \\ 2 & 2-\lambda_1 \end{vmatrix} &= 0 \\ (5-\lambda_1)(2-\lambda_1) - 2 \cdot 2 &= 0 \\ 10 - 5\lambda_1 - 2\lambda_1 + \lambda_1^2 - 4 &= 0 \\ \lambda_1^2 - 7\lambda_1 + 6 &= 0 \\ (\lambda_1 - 6)(\lambda_1 - 1) &= 0 \end{aligned}$$

thus $\lambda_1 = 6, \lambda_2 = 1$ are the two eigenvalue of Σ, which are the variances of the two principal components: 6 will be the variance of the first PC and 1 the variance of the second one.
To find the corresponding eigenvectors we comes back to our linear system:

$$\begin{bmatrix} 5-\lambda_1 & 2 \\ 2 & 2-\lambda_1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Now since $\lambda_1 = 6$ (consider first PC) we have

$$\begin{aligned} \begin{bmatrix} 1 & 2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{cases} -a_1 + 2a_2 = 0 \\ 2a_1 - 4a_2 = 0 \end{cases} \end{aligned}$$

In order now to solve this system of equations (eg say by substitution) we run in first problem: the second equation is the first one multiplied by -2: so we have a single equation with 2 unknown.
But up to now we didn't used the unit norm constraint, which become useful/comes into play, so we add to the first equation:

$$\begin{cases} -a_1 + 2a_2 = 0 \\ a_1^2 + a_2^2 = 1 \qquad (\mathbf{a}_1^\top \mathbf{a}_1 = 1) \end{cases}$$

Now we use substitution taking a_1 in the first equation

$$\begin{cases} a_1 = 2a_2 \\ 4a_2^2 + a_2^2 = 1 \end{cases} \quad \begin{cases} "" \\ 5a_2^2 = 1 \end{cases} \quad \begin{cases} "" \\ a_2^2 = \frac{1}{5} \end{cases} \quad \begin{cases} "" \\ a_2^2 = \pm\sqrt{\frac{1}{5}} \end{cases}$$

I *decide* to take the positive root so

$$\begin{cases} a_1 = \frac{2}{\sqrt{5}} \\ a_2 = \frac{1}{\sqrt{5}} \end{cases} \implies \mathbf{a}_1 = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}$$

This is my vector $\mathbf{a}_1$, the vector of coefficients of the first principal components. One could have either/also taken the negative root/solution obtaining

$$\begin{cases} a_1 = -\frac{2}{\sqrt{5}} \\ a_2 = -\frac{1}{\sqrt{5}} \end{cases}$$

These vectors are the same for our interest: the direction (pendenza) is the same. Thus *principal components are uniquely defined up to sign changes*: which can change from software to software (it depends on the software we use which solution is provided). Doing computation by hand, the positive valued vector is usually preferred
Let's calculate the second PC. Considering $\lambda_2 = 1$ we have

$$\begin{bmatrix} 5-1 & 2 \\ 2 & 2-1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$\begin{cases} 4a_1 + 2a_2 = 0 \\ 2a_1 + a_2 = 0 \end{cases}$$

Again we take one of the two equations only (the second) since are linear combinations and add the unity norm constraint

$$\begin{cases} 2a_1 + a_2 = 0 \\ a_1^2 + a_2^2 = 1 \end{cases} \quad \begin{cases} a_2 = -2a_1 \\ a_1^2 + 4a_1^2 = 1 \end{cases} \quad \begin{cases} "" \\ 5a_1^2 = 1 \end{cases} \quad \begin{cases} "" \\ a_1^2 = \pm\frac{1}{\sqrt{5}} \end{cases}$$

Taking the positive root we end with

$$\begin{cases} a_1 = \frac{1}{\sqrt{5}} \\ a_2 = -\frac{2}{\sqrt{5}} \end{cases} \implies \mathbf{a}_2 = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{bmatrix}$$

Note that while the vector of first component have all positive (or negative) elements if we want an orthogonal second component we need something that null the product of the two vectors so it will be of mixed signs (some elements will be positive and some negative).

Remark 41. R has two procedures (one based on spectral decomposition one based on Singular value decomposition) but eigenvector are the same.

Important remark 23. Let's appreciate two relevant aspects in the previous example:

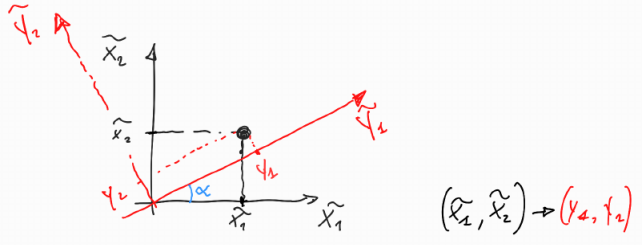


Figura 3.1: Rotation

1. the first eigenvector 6 is larger than any entries on the diagonal of original covariance matrix $\mathbf{\Sigma}$ (5 and 2). That is: the variance of the first principal component (6) is higher than any variance of the original variable.
   In general the variance of the first PC will *never* be smaller than the largest observed variance: it can be that the variance is equal but will be never be below;

2. looking at the trace of $\mathbf{\Sigma}$ it's 5+2; the sum of the eigenvalues is still 6+1=7. This always holds, so:

$$\sum_{i=1}^{p} \text{Var}\left[X_i\right] = \sum_{k=1}^{p} \lambda_k$$

   Remembering that the PCs are uncorrelated, we can write the variance covariance matrix of the PC as a matrix $\mathbf{\Lambda}$:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

   the off diagonal elements is always $= 0$ (being PCs uncorrelated). Thus we can rewrite

$$\text{Tr}\,\mathbf{\Sigma} = \text{Tr}\,\mathbf{\Lambda}$$

   So if one sum the eigenvalue variance of principal components it get the same variability of the original variable (trace of covariance matrix).
   Through PC we obtain a different way to split variability, we are not changing the total variability;

3. in general if first PC is a strict linear combinations (all entries with same sign), then the following (second and 3rd and so on) need/will be a contrast (different sign) in order orthogonality to be satisfied (vector product will be 0).

3.4 Interpretation

What does we do when we perform PC? 3.1 Let's assume variables are mean centered for simplicity and that we want to rotate the reference system.
The original point (x_1, x_2) will be transported in the new reference system and will be transformed to (y_1, y_2). Clearly the coordinates are linked and it turns

out that the coordinates on the new sistem depend on the angle between red and black line (α): if one change it, change the coordinates of the points.
Relationship between coordinates of two reference systems turns out to be:

$$\begin{cases} y_1 = x_1 \cos\alpha + x_2 \sin\alpha \\ y_2 = -x_1 \sin\alpha + x_2 \cos\alpha \end{cases}$$

We can collect the coefficient of the rotation/linear combination in

$$\mathbf{a}_1 = \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix}, \quad \mathbf{a}_2 = \begin{bmatrix} -\sin\alpha \\ \cos\alpha \end{bmatrix}$$

but $\cos^2\alpha + \sin^2\alpha = 1$ so both $\mathbf{a}_1, \mathbf{a}_2$ are unit norm vector. Moreover

$$\mathbf{a}_1^\top \mathbf{a}_2 = -\cos\alpha \sin\alpha + \sin\alpha \cos\alpha = 0$$

Important remark 24. So these vectors (depending on α) are orthogonal and with unit norm, as PCA's are. This means that:

- we can think PCA as an **orthogonal rotation of the reference system**;
- we have an infinity of possible orthogonal rotations (rotations are infinite, as *alpha*'s are): the one corresponding to PC is such that the **spread on the point of the first component is the maximum**;
- performing PC is just changing the reference system: actually we choose the optimal α with PC; trace remains the same because we're just rotating the system not changing data.

3.4.1 PCA and factor analysis

Important remark 25 (PCA vs FA). PCA is just a data transformation procedure (orthogonal rotation) and *is always feasible*: one can always move from X reference system to Y's one.
OTOH factor analysis has many things in common but *it is a model* and not necessarily *exists or is appropriate*.

Important remark 26 (PCA and factor analysis). Speaking about factor analysis we will see that there are connection with PC: PCA can be obtained using the same code used for fittining a factor analytics models, by putting some constraints. This is useful from programming pov because with one procedure we have two tools at our disposal.
However this caused a lot of confusion, people think that two methods are equivalent (thinking one of them is special case of other) so people used methods from factor analysis on PC (eg rotating principal components to improve interpretability, a thing is usually done in factor analyis): this thing drives prof crazy.
PC is an axis rotation done in order to maximize the variability explained. We've seen that PC solution is unique (but sign). This means that *if we rotate it to improve interpetability we have no more PC* (we obtain something that is easier to interpret but no longer the obtained first PCs optimize the explained variability).

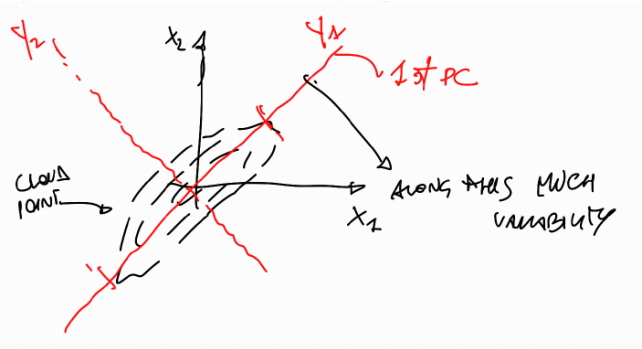


Figura 3.2: Pca popularity

3.4.2 Reason of popularity

Important remark 27 (Reason of popularity for PC). PC is popular because it allows us to collapse multidimensionality; eg in a situation like in figure 3.2 (mean centered for simplicity), along the first pc the variability is higher, along the second a bit less: the core of data is along Y_1, remaining part could be noise.

3.5 Features

3.5.1 Uncorrelation of components

Having used an orthogonal $\mathbf{a}_2$ we obtain something interesting, PCs have a very relevant statical property. Let's consider our:

$$\mathbf{\Sigma a}_2 = \lambda_2 \mathbf{a}_2$$

premultiplying by $\mathbf{a}_1^\top$

$$\mathbf{a}_1^\top \Sigma \mathbf{a}_2 = \lambda_2 \mathbf{a}_1^\top \mathbf{a}_2$$

but we know that for orthogonality constraint $\mathbf{a}_1^\top \mathbf{a}_2 = 0$ so

$$\mathbf{a}_1^\top \mathbf{\Sigma a}_2 = 0$$

what is this? This is the covariance between y_1 and y_2, the first two principal components. Let's prove it. Having

$$y_1 = \mathbf{a}_1^\top \mathbf{x}$$
$$y_2 = \mathbf{a}_2^\top \mathbf{x}$$

Thus the covariance between y_1 and y_2 is

$$\begin{aligned}
\text{Cov}\,(Y_1, Y_2) &= \mathbb{E}\left[(y_1 - \overline{y_1})(y_2 - \overline{y_2})\right] \\
&= \mathbb{E}\left[(\mathbf{a}_1^\top \mathbf{x} - \mathbf{a}_1^\top \boldsymbol{\mu})(\mathbf{a}_2^\top \mathbf{x} - \mathbf{a}_2^\top \boldsymbol{\mu})\right] \\
&= \mathbf{a}_1^\top \, \mathbb{E}\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top\right] \mathbf{a}_2 \\
&= \mathbf{a}_1^\top \mathbf{\Sigma a}_2
\end{aligned}$$

Remark 42. This means that first and second *principal components are uncorrelated* (covariance is numerator of correlation); the same holds for any pair of principal components.
This is a not scontated gift, under the constraint imposed. The fact that orthogonal vectors generate uncorrelated variable is not necessarily true: orthogonality and un-correlation are not synonims.[3]

Example 3.5.1. A counterexample. Consider the vectors of the canonic bases

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Any point in $\mathbb{R}^2$ can be obtained as combination of those two.
We're interested in a generic (x_1, x_2) we can think of coordinates as linear combination of vector from canonic base:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 = \begin{bmatrix} x_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Now considering any scatterplot of correlated variable (think of a rugby ball): each point on a cloud of correlated variables X_1 and X_2 can be generated by $\mathbf{e}_1, \mathbf{e}_2$; $\mathbf{e}_1$ and $\mathbf{e}_2$ are orthogonal but the corresponding linear combination are correlated.
The points can be thought of as linear combination with coefficients given by the vectors of the canonical basis (which are orthogonal by definition but the variables are correlated).
On the contrary on PC orthogonal vectors (eigen) and uncorrelated variables.

3.5.2 A and similarity of Σ and Λ

Remark 43. We can collect all eigenvector in a matrix called $\mathbf{A}$ (for p variables $\mathbf{A}$ is a $p \times p$ matrix), whose columns are the eigenvector of $\mathbf{\Sigma}$ ordered according to decreasing values of the corresponding eigenvalues $\mathbf{\Sigma}$

$$\underset{p\times p}{\mathbf{A}} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_p \end{bmatrix}$$

Example 3.5.2. In the example above will be

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \end{bmatrix}$$

Important remark 28 (Properties of $\mathbf{A}$). Matrix $\mathbf{A}$ has orthogonal columns and so

$$\mathbf{A}^\top \mathbf{A} = \mathbf{I}$$

this because eigenvector have unit norm (so in the diagonal $\mathbf{a}_i^\top \mathbf{a}_i = 1$) and are orthogonal (so out of main diagonal $\mathbf{a}_i^\top \mathbf{a}_j = 0$).

[3] As an exercise think a counterexample: an example in which you have linear combination of the observed variable that are identified by orthogonal vectors and are correlated

Definition 3.5.1 (Vector of principal components)**.** Once collected $\mathbf{A}$ we can write in matrix form the vector of principal components $\mathbf{y}$ as product of $\mathbf{A}$ and the vector of observed variables:

$$\underset{p\times 1}{\mathbf{y}} = \underset{p\times p}{\mathbf{A}^\top}\,\underset{p\times 1}{\mathbf{x}}$$

Important remark 29 (Similar matrices (reminder)). Two matrices $\mathbf{X}$ and $\mathbf{Y}$ are similar if there exists a non-singular matrix $\mathbf{Z}$ such that $\mathbf{Z}^{-1}\mathbf{Y}\mathbf{Z} = \mathbf{X}$.
Similar matrices share many properties: among them they have the same trace, that is $\operatorname{Tr}\boldsymbol{X} = \operatorname{Tr}\boldsymbol{Y}$.

It turns out that variance covariance of starting variables and of principal components are similar matrix. Infact we previously seen that:

1. the variance of a general multiple linear combinations $\mathbf{A}\mathbf{x}$ is

$$\operatorname{Var}[\mathbf{y}] = \mathbf{A}^\top \operatorname{Var}[\mathbf{x}]\,\mathbf{A} = \underbrace{\mathbf{A}^\top}_{p\times p}\underbrace{\boldsymbol{\Sigma}}_{p\times p}\underbrace{\mathbf{A}}_{p\times p}$$

2. the variance of principal components are λ_i the covariances are 0; if we want to write the variance covariance matrix of principal components, $\boldsymbol{\Lambda}$ it has eigenvalue in decreasing order in diagonal (variances) and 0 out of diagonal (covariances)

$$\underset{p\times p}{\boldsymbol{\Lambda}} = \begin{bmatrix} \lambda_1 & \dots & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & \lambda_i & \dots & 0 \\ \dots & & & & \\ 0 & \dots & 0 & \dots & \lambda_p \end{bmatrix}$$

3. thus the following equivalence have to hold

$$\operatorname{Var}[\mathbf{y}] = \mathbf{A}^\top\boldsymbol{\Sigma}\mathbf{A} = \boldsymbol{\Lambda}$$

considered that $\mathbf{A}^\top = \mathbf{A}^{-1}$ being $\mathbf{A}$ orthogonal, we can conclude that $\boldsymbol{\Sigma}$ and **$\boldsymbol{\Lambda}$ are similar**.

Thus:

- being similar they have the same trace. So $\operatorname{Tr}\boldsymbol{\Sigma} = \operatorname{Tr}\boldsymbol{\Lambda}$;
- this is the proof that sum of variances of observed variances coincides with sum of variances of principal components; in other words PC transform data but doesn't change *total variability*;
- actually what PCA does is to *apply the spectral decomposition to covariance matrix*: this is interesting from the statistical point of view in order to have uncorrelated new variables and same total variances (represented by eigenvectors).

3.5.3 Dependance on units of measure

Consider the following situations with two covariance matrix equal in all but in the units

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 90 & 50 \\ 50 & 90 \end{bmatrix}, \quad \mathbf{\Sigma}_2 = \begin{bmatrix} 9000 & 500 \\ 500 & 90 \end{bmatrix}$$

This is due to the fact that *data are the same* but in the first case X_1 is measured in cm, in the second case it is expressed in mm. Does the change in the measurement unit affect PCA results?
If we do spectral decomposition the following is obtained;

- for $\mathbf{\Sigma}_1$ the first principal component is:

  $$y_1 = 0.707x_1 + 0.707x_2$$

  and its corresponding variance/eigenvalue $\lambda_1 = 140$. As we can see:

  - the variables are equally weighted/important (coefficients 0.707) equally in the first component;
  - trace of covariance matrix is 180, while variance explained by first PC is 140. Considering the first PC only, we take into account 78% of total variability.

- for Σ_2 the first principal component is:

  $$y_1 = 0.998x_1 + 0.055x_2$$

  with $\lambda_1 = 9027.97$ and percentage explained is $\lambda_1 / \operatorname{Tr} \Sigma_2 = 99.32$ %. So in this case:

  - coefficients are very different: the combination is completely dominated by x_1, while x_2 has a small coeff
  - the explained variance increase just because of the unit measure and the increased importance of x_1 in the PC solution

Important remark 30. This example tells us that *PCA is not scale equivariant*: it depends on units of measurement. The solutions we obtain can be completely different.
This example has shown that it might be troublesome to use PCs on a covariance matrix when the variables are expressed according to different measurement scales, unless there is a strong conviction that the units of measurements chosen are the only ones that make sense.
Even when this condition holds, using the covariance matrix will not provide very informative PCs if the variables have widely differing variances, as the PCs will tend to reproduce those variables having the largest variance. Furthermore, with covariance matrices and non-commensurable variables the PC scores might be difficult to interpret.

Important remark 31. Often we don't want this to happen (that is our result to be dependent of the unit of measure); in that case standard practice is to **standardize before** applying PCA, which coincides to doing spectral decomposition on the correlation matrix $\boldsymbol{\rho}$ instead of the covariance $\mathbf{\Sigma}$ (we have seen that the covariance matrix of the standardized variables is the correlation matrix).

3.5.4 PCA on covariance vs correlation matrices

Remark 44. Is there a relationship between eigenvalues/vectors of $\mathbf{\Sigma}$ and $\boldsymbol{\rho}$ (correlation matrix in population)? If that is so we could move from one solutions to the other seamlessly.
It might seem that the PCs for a correlation matrix could be obtained fairly easily from those for the corresponding covariance matrix. However, this is not the case; the eigenvalues and eigenvectors of the correlation matrix have no simple relationship with those of the corresponding covariance matrix and there's no way to directly go from one solution to the other.

Let $\mathbf{\Delta}$ denote the diagonal matrix whose diagonal elements are the same as those of $\mathbf{\Sigma}$, that is

$$\mathbf{\Delta} = \begin{bmatrix} \sigma_1^2 & \dots & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & \sigma_i^2 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & 0 & \dots & \sigma_p^2 \end{bmatrix}$$

Then the correlation matrix $\boldsymbol{\rho}$ can be obtained from $\mathbf{\Sigma}$ (at the population level) as:

$$\boldsymbol{\rho} = \mathbf{\Delta}^{-1/2}\mathbf{\Sigma}\mathbf{\Delta}^{-1/2} \tag{3.3}$$

($\mathbf{S}$ is the sample analog of $\mathbf{\Sigma}$ while $\mathbf{D}$ is for $\mathbf{\Delta}$).
From 3.3 we can obtain a new way to write the $\mathbf{\Sigma}$ showing the link with correlation $\boldsymbol{\rho}$

$$\boldsymbol{\rho} = \mathbf{\Delta}^{-1/2}\mathbf{\Sigma}\mathbf{\Delta}^{-1/2} \implies \mathbf{\Sigma} = \mathbf{\Delta}^{1/2}\boldsymbol{\rho}\mathbf{\Delta}^{1/2} \tag{3.4}$$

Now

- if $\mathbf{a}$ is an eigenvector of $\boldsymbol{\rho}$ and λ is the corresponding eigenvalue, we have for the eigens relationship that:

$$\boldsymbol{\rho}\mathbf{a} = \lambda\mathbf{a}$$

- if $\mathbf{b}$ is an eigevector of $\mathbf{\Sigma}$ and δ is the corresponding eigenvalue, we have as well:

$$\mathbf{\Sigma}\mathbf{b} = \delta\mathbf{b}$$

- doing substitution in this last one using 3.4

$$\mathbf{\Sigma}\mathbf{b} = \delta\mathbf{b}$$
$$\mathbf{\Delta}^{1/2}\boldsymbol{\rho}\mathbf{\Delta}^{1/2}\mathbf{b} = \delta\mathbf{b}$$

  if we premultiply both sides to $\mathbf{\Delta}^{-1/2}$ we obtain

$$\boldsymbol{\rho}\mathbf{\Delta}^{1/2}\mathbf{b} = \delta\mathbf{\Delta}^{-1/2}\mathbf{b}$$

  now we decide/rename that $\mathbf{c} = \mathbf{\Delta}^{1/2}\mathbf{b}$ so

$$\boldsymbol{\rho}\mathbf{c} = \delta\mathbf{\Delta}^{-1}\mathbf{c} \tag{3.5}$$

Point is: is equation 3.5 an eigenvalue-eigenvector relationship for $\boldsymbol{\rho}$?

- $\mathbf{c}$ is eigengevector of $\boldsymbol{\rho}$ only if $\boldsymbol{\Delta}^{-1}\mathbf{c} = \mathbf{c}$, but $\boldsymbol{\Delta}^{-1}\mathbf{c} \neq \mathbf{c}$ because $\boldsymbol{\Delta} \neq \mathbf{I}$ (it's the matrix of variances of variables).
- So $\boldsymbol{\rho}\mathbf{c} = \delta\boldsymbol{\Delta}^{-1}\mathbf{c}$ is not an eigenvalue-eigenvector relationship.
  This tells us that the eigenvalues and eigenvectors of $\boldsymbol{\Sigma}$ are different from the eigenvalues/vectors of $\boldsymbol{\rho}$ and there is no way to move from one solutions to the other.

Important remark 32. Summarizing, NO, there's no relationship between eigenvalues/vectors of Σ and $\boldsymbol{\rho}$. When performing PCA we need to decide *before* to standardize or not: if we have

1. different units of measuremente or
2. very different variances

then standardize variables is better.
Otherwise if differences in variances are important and we don't want to homogeneize it, then use the covariance unstandardized matrix.

Example 3.5.3. Consider

$$\boldsymbol{\rho} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

derive principal components, what changes when ρ is positive or negative?

3.6 Number of components to choose

Important remark 33 (Need for criteria on number of components). In general with p variable we can end to p components.
However, especially if we want to collapse multidimensionality (as often is the case), we need *formal criteria* to choose the *number of components* which are enough to summarize the data. There are different answers to this which we deepen in the following after looking at some features/properties of PCA

Important remark 34 (Going to sample quantities). Our developments so far have been related to *random vectors*. When faced with a real situation, we can derive principal components starting from the sample correlation matrix $\mathbf{R}$ or from the sample covariance matrix $\mathbf{S}$, depending on whether variable standardization has been deemed necessary or not. We estimate $\boldsymbol{\Sigma}$ or $\boldsymbol{\rho}$ (in the population) with ML estimators:

- for $\boldsymbol{\Sigma}$ we use $\mathbf{S}$ (the ML estimates for $\boldsymbol{\Sigma}$, dividing codeviances by n). If we prefer we could use the unbiased estimate for $\boldsymbol{\Sigma}$ (dividing codeviances by $n-1$ instead of n); the variance of the principal components will be rescaled accordingly and the eigenvectors will not change;
- for $\boldsymbol{\rho}$ we use $\mathbf{R}$ where the same happens.

We will still denote by $\mathbf{a}_k$ the generic vector of coefficients and by l_k the corresponding eigenvalue, representing the variance of X_k . The properties derived for population PCs will hold for sample PCs as well.

Important remark 35 (Dimension reduction). Besides deriving uncorrelated linear combinations with maximum variance, a further goal of PCA is to perform *dimension reduction* i.e. to reduce a large number (p) of variables to a much smaller number (m) of PCs whilst retaining as much as possible of the variation in the p original variables.
Now we look into criteria to decide how many components we need/take to at the sample level.
How can we decide how many PCs we should keep in order to reduce dimensionality and at the same time preserve as much info as possible? There is a *tradeoff*: we want to reduce dimensionality but don't want to loose to much information.

Important remark 36. Different rule of thumb/heuristic rules (suggestion based on empirical evidence), not formal criteria, have been proposed to select the number m of principale components to keep:

1. percentage of explained/retained variance;
2. threshold on eigenvalue;
3. screeplot

For each one there are two "variants", working respectively with $\mathbf{S}$ or $\mathbf{R}$.

Important remark 37 (Why rule of thumbs). The reason why there's no formal criteria is because so far we made no assumption on distribution of variables: PC are data transformation and such can be always obtained (it's just an algebraic procedure done on covariance matrix or correlation matrix).
In principle we don't need info on population: as it's very general the only way we have to make decision is to use empirical rules (in essence it's a descriptive method).
Of course it is possible to define rigorous inferential procedures and tests for the selection of the number m of components that should be retained, but this would require distributional assumptions while the validity of the method lives independently of them.

3.6.1 Percentage of explained variance

Cumulative percentage of explained/retained variance is a first criterion and for what concerns:

- $\mathbf{S}$: we know that the

$$\operatorname{Tr}\mathbf{S} = \operatorname{Tr}\mathbf{L} = \sum_{i=1}^{p} s_i^2 = \sum_{k=1}^{p} l_k$$

  where $\mathbf{L}$ is the matrix with the eigenvalues (the sample analog of $\mathbf{\Lambda}$), while s_i^2 are the observed variances.
  The proportion of total variance explained by first PC is

$$\frac{l_1}{\operatorname{Tr}\mathbf{L}} \cdot 100$$

The proportion explained by first twos is

$$\frac{l_1 + l_2}{\operatorname{Tr} \mathbf{L}} \cdot 100$$

The percentage explained up to m components is

$$\frac{\sum_{k=1}^{m} l_k}{\operatorname{Tr} \mathbf{L}} \cdot 100$$

The criteria is to stop cumulating at the value m that make the above explained $\geq 80\%$

- **R**: similarly we have that

$$\operatorname{Tr} \mathbf{R} = \operatorname{Tr} \mathbf{L} = \sum_{i=1}^{p} 1 = \sum_{k=1}^{p} l_k = p$$

where we substituted the variance s_i^2 with 1, that is the correlation of each variable with itself. So with standardized data the total variance ($\operatorname{Tr} \mathbf{R}$) is just the number of variables p.
The reasoning is the same but percentage of explained by first component is now

$$\frac{l_1}{p} \cdot 100$$

And again the final criteria is:

$$\frac{\sum_{k=1}^{m} l_k}{p} \cdot 100 \geq 80\%$$

3.6.2 Threshold on eigenvalues (Kaiser's rule)

For the second rule, known as Kaiser's rule, we start seeing this applied correlation matrix (it was invented when working with standardized variable). For:

- **R**: we keep as many principal components as are the eigenvalues ≥ 1: in order to keep the principal components that do better than original variabiles (which have variance 1 being standardized), 1 act as a treshold to select the number of PCs;
- **S**: let's consider what 1 is for the correlation matrix above and find an equivalent for **S**. 1 is also the *average eigenvalue/variance* (sum of eigenvalues is p divided by number of elements p). Therefore we can translate the Kaiser's rule in the non standardized world by keeping as many PCs as are the eigenvalues of **S** which are $\geq$ than the mean variance/eigenvalue:

$$\bar{l} = \frac{\sum_{k=1}^{p} l_k}{p}$$

Important remark 38 (Joliffe's variant). In some books a slightly modified version of the rule above (due to Joliffe) is presented:

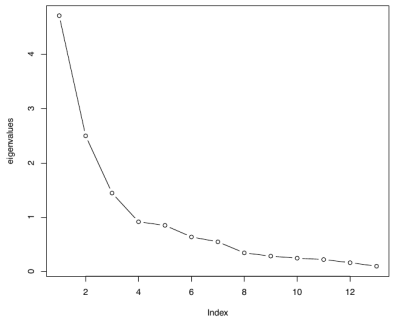


Figura 3.3: Screeplot

- for **R**: considering that I'm working in a sample if I reject eigenvalues below 1/mean it may be that I'm refusing something that in the population is above 1/mean and the sample was "particular".
  In other words, variability slightly lower than 1 can be kept/retained if in the population is actually 1 or more.
  After some simulation Joliffe ended with choosing to lower the sample treshold to 0.7 (instead of 1);
- in the case of unstandardized data we use $0.7 \cdot \bar{l}$

3.6.3 Screeplot

It's a graphical rule common to both matrix **S** and **R**. We plot the eigenvalues in order from the highest to the lower (that is we plot l_k against k, with $k = 1, \ldots, p$) to describe the fall/slope .
A typical pattern is shown in fig 3.3: after a sharp decline the curve tends to become flat. The flat portion corresponds to noise components, unable to capture the leading variability; the rule therefore suggests that m should correspond to the value of k at which the elbow of the scree plot occurs (3 in the picture).

3.7 PC interpretation

How do we interpret principal components? Assume we have two variables (height and weight) and that the variables are mean centered (just to ease graph without losing generality). There are **two strategies** to do it:

- one way we do by looking at coefficients/element of eigenvector (we study sign and the value of these elements) at figuring out the *extremes* of the derived linear combination.
  Important: remember that eigenvector has unit norm: if norm is unit we can measure the importance of each variable for a component and look how variables balance in the vector (this is the prof favorite method).
  Most of the case first PC have the same sign on all elements of the eigenvector (not always, but often is like that), while the other are contrasts.
- otherwise we look at the *correlation* between each principal component and the observed variables

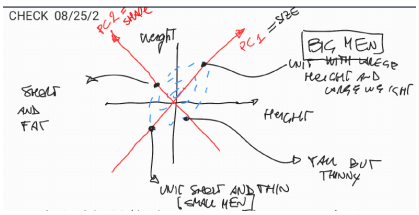


Figura 3.4: PCA interpretation using extremes

3.7.1 Extremes interpretation

Strategy is to look at extremes and name the pc according to these extremes; the method is called *reification of PC* (give meaning to math construct from an empirical point of view).
Looking at fig 3.4 if we project the point:

- into the first direction we're able to order individuals on the "size" variable: looking at the first component at one end we have mes with high height/weight (BIG men) at the other low height/weight (SMALL men);
- along the second orthogonal direction we have that are characterizing individual according to the "shape": on one hand we have short and fat men, on the other tall but thin

Example 3.7.1. A popular example with 6 variables on chicken (*galline bianche sperone*, white leghorn chicken) which are:

NB: esempio proiettato e poi non condiviso su virtuale

- length cranium (head)
- width cranium (head)
- length humerus (wing)
- length ulna (wing)
- length femur (legs)
- length of tibia (legs)

with 6 variables we have up to 6 PC: the example is famous because to all 6 components we can give a meaning! (usually is not that so).
Data available is on correlation matrix (we have a $6x6$ matrix only): with R's `eigen` we calculates eigenvalues and vectors eigenvalues:

- the first PC accounts for 76%, while the first two we have more than 80% (so 2 are enough according to proportion of explained variance).
- applying Kaiser's rule we would conclude that 1 PC is enough; to lower the bound according to Joliffe we take two PC (second has eigenvector 0.71)

Our focus is now interpreting: each column corresponds to an eigenvector, plotted in the order of the corresponding eigenvalues so vectors matrix is our **A**

1. the first eigenvector is composed of all negative coefficients (could be positive either): looking at elements of the vector (their absolute value) they are more or less the same (between 0.3 and 0.4): to interpret (assume is positive to ease) what a chicken with an high value on the first principal components looks like?

$$Y_1 \cong 0.3(x_1 + x_2) + 0.4(x_3 + x_4 + x_5 + x_3)$$

a chicken with large first PC1 will be big while small values will correspond to small chicken; so PC1 describes the *size* of the chicken;

2. the second is a contrast (not all are the same) otherwise they are not orthogonal: we have positive values with two variables describing the head and negative values for wings and legs. I need a large value of positive. High score on the second are for chicken with large head and small body, low scores small head and big body.

$$Y_2 \approx 0.6(x_1 + x_2) - 0.2(x_3 + ... + x_6)$$

so PC2 can be seen as a "shape" (large/small head vs body).
These are the main two components; most variability of the chicken are related to size and shape

3. if we go on and consider the third PC: values with body characteristics are really near to 0 so I can ignore this, only the head variables is important

$$y_3 \approx 0.77x_1 - 0.63x_2$$

I have again a contrast (because of the sign): on one side chicken with long head and not large, on the other side short and large head. So this third PC describes "head shapes"

4. the fourth PC neglect/ignore the head data. We focus on legs and wings

$$y_4 = -0.5(x_3 + x_4) + 0.5(x_5 + x_6)$$

large positive values correspond to chicken with big legs and small wings, on the other side large wings and small legs (another measure of shape of the body)

5. and so on ...

Example 3.7.2 (Jolicoeur and Mosimann (1960))**.** Jolicoeur and Mosimann (1960) measured carapace length, width and height (in mm) for 24 male turtles:

- the average vector for the three variables is $\overline{x} = (113.375, 88.29167, 40.70833)$
- the variances are 132.9844, 47.9566, 10.78993 respectively.

The variances, however expressed in the same measurement unit, are very different. This suggests that a kind of normalization is advisable in order to prevent the first PC from being completely dominated by the first variable.

They used a log transform the following covariance matrix obtained from the log transformed data:

$$\mathbf{S} = 10^{-3} \begin{bmatrix} 11.072 & 8.019 & 8.160 \\ 8.019 & 6.417 & 6.005 \\ 8.160 & 6.005 & 6.773 \end{bmatrix}$$

with total variance 0.024261488.
The log-transformation effectively homogenized variances in this case. It cannot be suggested as a general rule because it requires that all the data are positive and no zero value is present in the data set, and this is not always the case. Anyway, for reasons that will be made clear in the following, it is very useful in studies of size and shape (allometry) as is the one at hand.
The eigenvalues, i.e. the variances of the principal components are $l_1 = 0.023303$, $l_2 = 0.000598$, $l_3 = 0.00036$.
The first principal component accounts for the 96.0508 per cent of the total variance and so it is enough, alone, to reproduce most of the observed variability. It is also evident that its eigenvalue is larger than the average eigenvalue. The modified version of Kaiser's rule still suggests one PC is enough.

- The first eigenvector is $\mathbf{a}_1^\top = (0.683102, 0.510220, 0.522539)$. The first principal component is a linear combination of the log transformed carapace measurements; it separates large turtles, with a large, long, thick carapace, and small turtles, with a narrow, short, thin carapace. We might say it separates turtles according to their size. Moreover, since

$$\begin{aligned} y_1 &= 0.683 \ln(length) + 0.510 \ln(width) + 0.523 \ln(height) \\ &= \ln\left[(lenght)^{0.683} (width)^{0.510} (height)^{0.523}\right] \end{aligned}$$

  the first PC may be viewed as the $\ln(volume)$ of a box whose dimensions have somehow been adjusted in order to account for the round shape of the carapace.

- The second eigenvector is $\mathbf{a}_2^\top = (-0.159479, -0.594012, 0.78849)$. Even if the second PC is not important in recovering a reduced dimension representation of our data we ca try to give it an empirical meaning. It is a linear contrast i.e. some variables have a positive coefficient and some have a negative one. It separates turtles having a thick but small carapace, from turtles having a thin and large carapace. It can be thought of as a measure of *shape*.

3.7.2 Correlation between PC and variables

Remark 45. Some authors suggest to use the correlations between each principal component and all the observed variables instead. In order to derive a simple expression for these correlations let's restate the problem in terms of random variables/vectors.

What is the correlation between each principal component and the observed variables? We're interested in $cov(y_1, \mathbf{x})$ with y_1 is a scalar random variable (1×1), $\mathbf{x}$ is a vector of observed variable ($p \times 1$ vector).

If we assume without loss of generality to work with *mean centered* observation $\mathbf{x}$, the covariance is

$$\begin{aligned}\text{Cov}(y_1, \mathbf{x}) &= \mathbb{E}[y_1 \mathbf{x}] \overset{(1)}{=} \mathbb{E}\left[\mathbf{a}_1^\top \mathbf{x}\mathbf{x}^\top\right] \overset{(2)}{=} \mathbf{a}_1^\top \mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right] \\ &= \mathbf{a}_1^\top \boldsymbol{\Sigma} = \lambda_1 \mathbf{a}_1^\top\end{aligned}$$

where in

- (1) we just replaced the first principal component.
- (2) $\mathbf{a}_1$ is a constant vector that can be taken out of expectation

Thus the covariance between y_1 and $\mathbf{x}$ $\mathbf{a}_1^\top \boldsymbol{\Sigma} = \lambda_1 \mathbf{a}_1^\top$. The correlation instead is covariance divided by the standard deviation and in our case is

$$\begin{aligned}\text{Corr}(y_1, \mathbf{x}) &= \frac{1}{sd(y_1)} \cdot \text{Cov}(y_1, \mathbf{x}) \cdot \boldsymbol{\Delta}^{-1/2} \\ &= \frac{1}{\sqrt{\lambda_1}} \lambda_1 \mathbf{a}_1^\top \boldsymbol{\Delta}^{-1/2} \\ &= \sqrt{\lambda_1} \mathbf{a}_1^\top \boldsymbol{\Delta}^{-1/2}\end{aligned}$$

where

- (1) we needed to divide by the stanadard deviation of $\mathbf{x}$ (using the diagonal matrix Δ containing just the variances).
- (2) to define the standard deviation of first PC, $sd(y_i)$, it's the $\sqrt{\text{Var}[y_i]} = \sqrt{\lambda_1}$

In the sample the corresponding formula to adopt is:

$$\text{Corr}(y_1, \mathbf{x}) = \sqrt{l_1} \mathbf{a}_1^\top \mathbf{D}^{-1/2}$$

Remark 46. Correlation is a function of vector $\mathbf{a}_1^\top$ but actually here we just work with pairwise correlations of principal components with starting variables (so we loose the multivariate feelings/grip).

Example 3.7.3 (Jolicoeur and Mosimann (1960))**.** In the example above $\text{Corr}(y_1, \mathbf{x}) = (0.99, 0.97, 0.97)$ that is the first principal component is highly positively correlated with each of the observed variables. All variables contribute more or less in the same way to the first principal component, confirming the interpretation given above.

3.8 Principal component scores and usage

Scores are nothing but values of each components y:

- $\mathbf{Y}$ is $n \times p$ matrix which contains the coordinates for the n units in the new space spanned by the p principal components; it's obtained as

$$\underset{n \times p}{\mathbf{Y}} = \underset{n \times p}{\mathbf{X}} \underset{p \times p}{\mathbf{A}}$$

  where $\mathbf{X}$ is original data matrix and $\mathbf{A}$ is the matrix of the eigevectors.

- if we don't want to keep all the PCs but only m we'll have

$$\underset{n\times m}{\mathbf{Y}_m} = \underset{n\times p}{\mathbf{X}}\,\underset{p\times m}{\mathbf{A}_m}$$

where $\mathbf{A}_m$ is the matrix containing the first m eigenvectors.

3.8.1 Plotting

If $m = 2$ the scores can be plotted on a Cartesian plane: as this bi-dimensional plot is an approximation of the structure in the original p-dimensional space, it may happen that not all of the n statistical units are well represented in the bi-dimensional projection.

A way to check the adequacy of the approximation for each statistical unit is an inspection of the scores on the last (discarded) PCs, which actually are useful in at least two situations.

3.8.2 Goodness of first PCs

If a unit has high scores on the last PCs this means that it is largely displaced when projected on the PC plane and therefore it is misrepresented.

Example 3.8.1. Assume we performed PC in this room (x, y, z of object in space): two points/components are enough to project students head/position on the floor but if we project the lamp too it will be near the students over it is (despite being distant from them in 3d space).
How can we measure the *quality of representation*: if there's a unit that is very distant (lamp) from the other how can we say this? It turns out that the *coordinates along the last PC are a measure the quality of the first PCs.*

Important remark 39. One could consider, for each statistical unit the sum of the squared scores on the last $p-m$ PCs. The position on the PC plane of those units having a large value of such a sum should be carefully considered.
I can measure the distance of each units from the projection using only the first m PCs: this below is a measure of the quality of the m dimensional representation (using only the first m principal components) of unit j.

$$y_{j,m+1}^2 + y_{j,m+2}^2 + \ldots + y_{j,p}^2$$

Remark 47. If the measure above is much larger than its average, with respect to all the units, then this means that unit j is badly represented by m PCs alone.

3.8.3 Multicollinearity

Another possible use of last PC is in multicollinearity diagnosis (for linear regression), other than methods such as VIF.
Multicollinearity means in regression that in the $\mathbf{X}$ we have one column is almost a combination of other vectors. If this is so, we can use PC to spot it: if there are eigenvalue close to 0 there's multicollinearity/almost constant linear relationships among the covariates.

3.9 PCA and SVD

SVD is a matrix decomposition, way to decompose matrix $\mathbf{X}$, that tells us that given an $n \times p$ matrix $\mathbf{X}$, we can decompose it it in the product of three matrices

$$\underset{n\times p}{\mathbf{X}} = \underset{n\times p}{\mathbf{U}}\underset{p\times p}{\mathbf{P}}\underset{p\times p}{\mathbf{V}^\top}$$

where:

- $\mathbf{X}$ is the data matrix;
- $\mathbf{U}$ is the **matrix of the left singular vectors**, that is the eigenvectors of $\mathbf{X}\mathbf{X}^\top$ corresponding to non 0 eigenvalues. Note that $\mathbf{X}\mathbf{X}^\top$ is not we used so far ($\mathbf{X}^\top\mathbf{X}$) $\mathbf{X}\mathbf{X}^\top$ is $n \times n$ and has n dimensional eigengevectors. Keeping eigenvector corresponding to non-null eigenvalues we'll have a $n \times p$ matrix;
- $\mathbf{P}$ is the **matrix of the singular values**, a square diagonal matrix whose diagonal entries are the square root of the non zero eigenvalues of $\mathbf{X}\mathbf{X}^\top$ (or $\mathbf{X}^\top\mathbf{X}$, as the two matrices have the same eigenvalues);
- $\mathbf{V}$ is the **matrix of right singular vectors** (i.e. the eigenvectors of $\mathbf{X}^\top\mathbf{X}$).

Now let's assume we work on mean centered data and consider $\tilde{\mathbf{X}}$; the codeviance matrix (numerator of covariance matrix) can be written as product of n times the covariance matrix

$$\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} = n\mathbf{S}$$

From matrix algebra it turns out that:

- $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ and $\mathbf{S}$ have the *same eigenvectors* (what changes is only eigenvalues). From PCA, we know that eigenvectors of $\mathbf{S}$ are the elements of matrix $\mathbf{A}$ (solution to the PC problems): so $\mathbf{A}$ will be the matrix of the eigenvectors of both $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ and $\mathbf{S}$.
- the eigenvalues of $\mathbf{S}$ will be the eigenvalues of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ divided by n

Thus if we consider the singular value decomposition of $\tilde{\mathbf{X}}$ we have

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{P}\mathbf{A}^\top$$

going back to the PC scores $\mathbf{Y}$, we can replace $\tilde{\mathbf{X}}$ with its SVD:

$$\mathbf{Y} = \tilde{\mathbf{X}}^\top\mathbf{A} = \mathbf{U}\mathbf{P}\mathbf{A}^\top\mathbf{A} \overset{(1)}{=} \mathbf{U}\mathbf{P}$$

where in (1) since $\mathbf{A}$ is orthogonal so $\mathbf{A}^\top\mathbf{A} = \mathbf{I}$.

Important remark 40. So using SVD we obtained all the main elements of PC analysis:

- $\mathbf{A}$ (of PCs) is $\mathbf{V}$ (in the Svd decomposition)
- $\mathbf{UP}$ is the matrix with the PC scores $\mathbf{Y}$ (where the diagonal matrix $\mathbf{P}$ is $\mathbf{P} = n^{1/2}\mathbf{L}^{1/2}$)

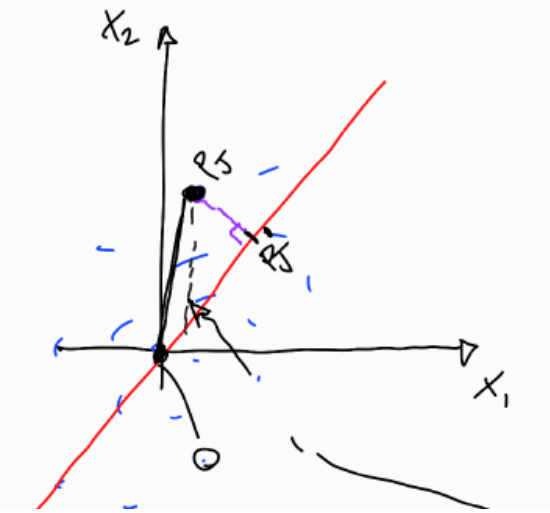


Figura 3.5: Orthogonal least squares

So we can obtain PC either by spectral decomposition or by SVD of centered data matrix.

Important remark 41. Moreover if we consider only m principal components then we will have that:

$$\tilde{\mathbf{X}} \approxeq \mathbf{U}_m \mathbf{P}_m \mathbf{V}_m^\top$$

There's a theorem due to Eckart and Young that states that the equation above is the best rank m approximation of $\tilde{\mathbf{X}}$.
If we have $\tilde{\mathbf{X}}$, which is of rank p, and we want to approximate it with a lower rank matrix having rank m, with $m < p$, then there's no other m approximation better than this. We just consider the first m singular values and corresponding left and right singular vectors.

Remark 48. YouTube: SVD song it had to be U - the SVD song statistical song youtube michael greenacree

3.10 Historical remarks (orthogonal least square)

Remark 49. When performed regression usually we wanted to minimize the sum of square of black lines (fig 3.5).
PC was first invented by Pearson in 1901: he wanted to find the line which minimize the sum of the squared *orthogonal distances* (violet) between points and the line. For this reason methods is also known as orthogonal least square. This was a way to take into account possible measurement errors in the random variable x (while in regression analysis this is considered fixed/without errors).

Let P_j be a point, O the origin and P_j' being the orthogonal projection of P_j along the red line; we can join P_j the origin obtaining a rectangle triangle. By Pithagora's theorem this holds:

$$\overline{OP_j}^2 = \overline{P_j P_j'}^2 + \overline{OP_j'}^2$$

Summing with respect to all the units (being j is the generic units)

$$\sum_{j=1}^{n} \overline{OP_j}^2 = \sum_{j=1}^{n} \overline{P_j P'_j}^2 + \sum_{j=1}^{n} \overline{OP'_j}^2$$

we see the terms

- the overall total sum of square $TSS = \sum_{j=1}^{n} \overline{OP_j}^2$ is fixed (and does not depend on the reference system): if working with mean centered data this is the deviation from the mean
- the quantity $\sum_{j=1}^{n} \overline{P_j P'_j}^2$ is the quantity that Pearson wanted to minimize
- the quantity $\sum_{j=1}^{n} \overline{OP'_j}^2$, is the sum of square along the direction/red line, which if the other is minimized (being TSS fixed) it turns out to be maximized in the process.
  Point is that the third element is the *variability of the first PC component*, indeed

  $$\sum_{j=1}^{n} \overline{OP'_j}^2 = \sum_{j=1}^{n} (y_{j1} - \overline{y}_1)^2 = Dev(y_1)$$

  So solving Pearson's problem amounts to find the line/direction along which the TSS along the direction is maximum and this coincides with solving Hotelling's problem.

3.11 Lab

We do PCA both manually and results are different from functions.
Generally, PCA aims to find the linear combinations of the observed variables which explain as much of the data variability as possible. Besides, this method is implemented to perform dimension reduction, i.e. to produce a smaller set of uncorrelated variables from the larger set of correlated ones.

3.11.1 Example 1: Job performance

Data contain observations on fifty police officers that were rated by their supervisors in 6 categories as part of standard police departmental administrative procedure:

- `commun`: Communication Skills
- `probl_solv`: Problem Solving
- `logical`: Logical Ability
- `learn`: Learning Ability
- `physical`: Physical Ability
- `appearance`: Appearance

```
job <- read.table("data/job_perf.txt", header = TRUE)
head(job) # 6 var
```

```
##   commun probl_solv logical learn physical appearance
## 1     12         52      20    44       48         16
## 2     12         57      25    45       50         16
## 3     12         54      21    45       50         16
## 4     13         52      21    46       51         17
## 5     14         54      24    46       51         17
## 6     14         48      20    47       51         18
```

```
summary(job)
```

```
##      commun        probl_solv       logical          learn      
##  Min.   :12.00   Min.   :48.00   Min.   :20.00   Min.   :44.00  
##  1st Qu.:16.00   1st Qu.:52.25   1st Qu.:22.00   1st Qu.:48.00  
##  Median :18.00   Median :54.00   Median :24.00   Median :50.00  
##  Mean   :17.68   Mean   :54.16   Mean   :24.02   Mean   :50.28  
##  3rd Qu.:19.75   3rd Qu.:56.00   3rd Qu.:26.00   3rd Qu.:52.00  
##  Max.   :24.00   Max.   :59.00   Max.   :31.00   Max.   :56.00  
##     physical       appearance   
##  Min.   :48.00   Min.   :16.00  
##  1st Qu.:52.25   1st Qu.:19.00  
##  Median :54.00   Median :21.00  
##  Mean   :54.16   Mean   :21.06  
##  3rd Qu.:56.00   3rd Qu.:23.00  
##  Max.   :59.00   Max.   :28.00
```

```
## to perform PCA manually we use matrix computation
X <- as.matrix(job) # first transform to matrix
n <- nrow(job)      # we need number of obs in our dataset

# we should check that variables are correlated prior to performing
cor(X)
```

```
##                commun probl_solv   logical     learn  physical appearance
## commun     1.0000000  0.1313258 0.2252998 0.9821863 0.9767974  0.9692466
## probl_solv 0.1313258  1.0000000 0.3637625 0.1425815 0.1640910  0.1287805
## logical    0.2252998  0.3637625 1.0000000 0.2307109 0.2582155  0.2164945
## learn      0.9821863  0.1425815 0.2307109 1.0000000 0.9814717  0.9718918
## physical   0.9767974  0.1640910 0.2582155 0.9814717 1.0000000  0.9690222
## appearance 0.9692466  0.1287805 0.2164945 0.9718918 0.9690222  1.0000000
```

Is PCA justified? It is possible to reduce data dimensionality without losing too much information only if the original variables are highly correlated. In fact, when the correlation between variables is weak, a larger number of components is needed in order to capture enough variability.

Is PCA justified? Here yes, Since the correlation values look quite large (eg `learning` and `phisical` and `appearance` with `commun`), in this context PCA

seems to be useful in order to reduce dimensionality.

Should the PCA be carried out with the covariance or the correlation matrix? To choose between the covariance and the correlation matrix is like to choose whether to work with standardized variables or not. Indeed, the correlation matrix of X coincides with the covariance matrix if the variables are standardized.

The use of correlation matrices (that is standardized variables) is recommended when the magnitudes of the variables are very different. Indeed, the latter implies different variance magnitudes, which would imply that the first few principal components are dominated by the variables with bigger magnitudes. Standardization gives to all the variables the same importance in determining the components. Since in this example the variances are quite similar, it is possible to use the covariance matrix to carry out PCA.

```
## we need to chose if work with standardized data or not. To choose it we can
## look at variances of variables. Either using diag(S) or using apply/lapply

apply(X, 2, var)

##     commun probl_solv    logical      learn   physical appearance
##   7.528163   5.810612   6.183265   8.042449   5.810612   8.955510

# var have the similar variances it's not needed to work with standardized
# data. For this dataset we can work with covariance matrix
## to compute PCA do spectral decomposition (using eigen function)
```

3.11.1.1 PCA using matrix functions

In order to perform PCA we need to do spectral value decomposition of covariance/correlation matrix: here we chose to do it on covariance matrix. So the decomposition become

$$\mathbf{A}^\top \mathbf{S} \mathbf{A} = \mathbf{L}$$

where

- $\mathbf{L}$ diagonal matrix of eigenvalues
- $\mathbf{A}$: orthonormal matrix of eigenvectors

In R they can be found using `eigen` function

```
## start with the centering matrix
C <- diag(n) - 1/n * rep(1, n) %*% t(rep(1, n)) # centering matrix
Xc <- C %*% X  #centered data
round(colMeans(Xc), 4) # check its' approx 0

##     commun probl_solv    logical      learn   physical appearance
##          0          0          0          0          0          0
```

```
S <- t(Xc) %*% Xc / n  #covariance matrix
S

##            commun probl_solv logical  learn physical appearance
## commun     7.3776     0.8512  1.5064 7.4896   6.3312     7.7992
## probl_solv 0.8512     5.6944  2.1368 0.9552   0.9344     0.9104
## logical    1.5064     2.1368  6.0596 1.5944   1.5168     1.5788
## learn      7.4896     0.9552  1.5944 7.8816   6.5752     8.0832
## physical   6.3312     0.9344  1.5168 6.5752   5.6944     6.8504
## appearance 7.7992     0.9104  1.5788 8.0832   6.8504     8.7764

## Doing PCA by matrix functions
spectral <- eigen(S)
spectral$values

## [1] 29.7582011  7.4672075  3.7076488  0.2868423  0.1467846  0.1173156

L <- diag(spectral$values)
A <- spectral$vectors
rownames(A) <- rownames(S)
A

##                   [,1]        [,2]        [,3]         [,4]        [,5]
## commun     -0.49155807  0.08748571  0.01214618 -0.380639251  0.75573635
## probl_solv -0.08675503 -0.69046582  0.71781729  0.008879732  0.01827673
## logical    -0.13662756 -0.70496706 -0.69539380  0.015845890  0.01597971
## learn      -0.50947274  0.08031174  0.02069595 -0.329958740 -0.33064205
## physical   -0.43261611  0.04023081  0.01036964 -0.217845082 -0.56076816
## appearance -0.53428266  0.10274312  0.02196412  0.835735944  0.06699316
##                    [,6]
## commun     -0.185864772
## probl_solv  0.007480667
## logical     0.016594522
## learn       0.717887386
## physical   -0.670223722
## appearance  0.023681484
```

`spectral` returns a list of two objects: `values` (eigenvalues), and `vectors` (matrix of eigenvectors).
Now we can:

1. look at variablity explained by PC (looking at eigenvalue):

```
spectral$value / sum(spectral$value) # percentage explained by each component

## [1] 0.717341652 0.180002109 0.089375394 0.006914529 0.003538342 0.002827973

cumsum(spectral$value) / sum(spectral$value) # cumulative percentage explained

## [1] 0.7173417 0.8973438 0.9867192 0.9936337 0.9971720 1.0000000
```

first PC explain 72% while the second the 18%, the last for a small percentage the cumulative; so we can stop at two components (90% of variability)

2. compute the total variance explained by the components

```
sum(spectral$values) # sum of eigenvectors coincides with

## [1] 41.484

sum(diag(S))         # sum of variances (total variability)

## [1] 41.484
```

3. look at **A** this is the loading matrix (rows variables and columns principal components, we can try to interpret)

```
round(A, 3)

##              [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## commun     -0.492  0.087  0.012 -0.381  0.756 -0.186
## probl_solv -0.087 -0.690  0.718  0.009  0.018  0.007
## logical    -0.137 -0.705 -0.695  0.016  0.016  0.017
## learn      -0.509  0.080  0.021 -0.330 -0.331  0.718
## physical   -0.433  0.040  0.010 -0.218 -0.561 -0.670
## appearance -0.534  0.103  0.022  0.836  0.067  0.024
```

4. compute the scores

```
head(Y <- X %*% A) # matrix with same dimenson of our original dataset

##           [,1]      [,2]     [,3]      [,4]      [,5]      [,6]
## [1,] -64.87341 -41.84505 25.32417 -15.39198 -30.05441 -1.714282
## [2,] -67.36502 -48.66144 25.47772 -16.03400 -31.33531 -2.216466
## [3,] -66.55825 -43.77018 26.10584 -16.12402 -31.45406 -2.305286
## [4,] -68.35267 -42.07847 24.73539 -16.23449 -31.55929 -2.434767
## [5,] -69.42762 -45.48682 24.09698 -16.54983 -30.71906 -2.555887
## [6,] -69.40433 -38.34110 22.61432 -16.16072 -31.15629 -1.925580
```

for each obs the column are the principal components; when we select a lower number of components we reduce the number of variables

3.11.1.2 Using R functions

To perform principal component analysis there are also functions already implemented in R. These are:

- `prcomp`, that uses function svd on the data matrix.
- `princomp`, that uses function eigen to compute the decomposition of S/R, as we just did

`prcomp` `prcomp` perform PCA using SVD. When using `prcomp`

- `x` is our original dataset (matrix or data.frame); by default the functions center the data
- `scale.` to be set to `TRUE` if we want to work with standardized data too by dividing by (unbiased) standard deviation (default = `FALSE`)

To perform PCA

```
(prcomp_job <- prcomp(X, scale. = FALSE))

## Standard deviations (1, .., p=6):
## [1] 5.5104910 2.7603622 1.9450746 0.5410141 0.3870144 0.3459911
##
## Rotation (n x k) = (6 x 6):
##                   PC1         PC2         PC3          PC4         PC5
## commun     0.49155807 -0.08748571 -0.01214618  0.380639251 -0.75573635
## probl_solv 0.08675503  0.69046582 -0.71781729 -0.008879732 -0.01827673
## logical    0.13662756  0.70496706  0.69539380 -0.015845890 -0.01597971
## learn      0.50947274 -0.08031174 -0.02069595  0.329958740  0.33064205
## physical   0.43261611 -0.04023081 -0.01036964  0.217845082  0.56076816
## appearance 0.53428266 -0.10274312 -0.02196412 -0.835735944 -0.06699316
##                     PC6
## commun     -0.185864772
## probl_solv  0.007480667
## logical     0.016594522
## learn       0.717887386
## physical   -0.670223722
## appearance  0.023681484

# summary to obtain further prop of variance explained and cumulative

summary(prcomp_job)

## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     5.5105 2.7604 1.94507 0.54101 0.38701 0.34599
## Proportion of Variance 0.7173 0.1800 0.08938 0.00691 0.00354 0.00283
## Cumulative Proportion  0.7173 0.8973 0.98672 0.99363 0.99717 1.00000
```

returned `prcomp_job` object has:

- `sdev`: square root eigenvalue. Looking at eigenvalues

```
prcomp_job$sdev^2    # eigenvalues

## [1] 30.3655113  7.6195995  3.7833151  0.2926963  0.1497802  0.1197098

spectral$values      # they're a bit different

## [1] 29.7582011  7.4672075  3.7076488  0.2868423  0.1467846  0.1173156
```

Eigenvalues are a bit different due to the fact that we computed starting from the biased covariance matrix while *prcomp* use the unbiased covariance matrix

```
spectral$values * n / (n - 1) # exact same results as prcomp
```

```
## [1] 30.3655113  7.6195995  3.7833151  0.2926963  0.1497802  0.1197098
```

- `rotations`: matrix eigenvectors. Results are the same of handmade up to sign changes (in absolute values they're exactly the same)

```
prcomp_job$rotation
```

```
##                   PC1         PC2         PC3          PC4         PC5
## commun     0.49155807 -0.08748571 -0.01214618  0.380639251 -0.75573635
## probl_solv 0.08675503  0.69046582 -0.71781729 -0.008879732 -0.01827673
## logical    0.13662756  0.70496706  0.69539380 -0.015845890 -0.01597971
## learn      0.50947274 -0.08031174 -0.02069595  0.329958740  0.33064205
## physical   0.43261611 -0.04023081 -0.01036964  0.217845082  0.56076816
## appearance 0.53428266 -0.10274312 -0.02196412 -0.835735944 -0.06699316
##                    PC6
## commun     -0.185864772
## probl_solv  0.007480667
## logical     0.016594522
## learn       0.717887386
## physical   -0.670223722
## appearance  0.023681484
```

```
A
```

```
##                   [,1]        [,2]        [,3]         [,4]        [,5]
## commun     -0.49155807  0.08748571  0.01214618 -0.380639251  0.75573635
## probl_solv -0.08675503 -0.69046582  0.71781729  0.008879732  0.01827673
## logical    -0.13662756 -0.70496706 -0.69539380  0.015845890  0.01597971
## learn      -0.50947274  0.08031174  0.02069595 -0.329958740 -0.33064205
## physical   -0.43261611  0.04023081  0.01036964 -0.217845082 -0.56076816
## appearance -0.53428266  0.10274312  0.02196412  0.835735944  0.06699316
##                    [,6]
## commun     -0.185864772
## probl_solv  0.007480667
## logical     0.016594522
## learn       0.717887386
## physical   -0.670223722
## appearance  0.023681484
```

- `x`: it's the matrix of scores $\mathbf{Y}$.

```
## compare with handmade
head(prcomp_job$x)

##             PC1        PC2        PC3        PC4        PC5         PC6
## [1,] -12.096558 -2.5563952 -0.8710214 -1.2643930 -0.7954799  0.47326072
## [2,]  -9.604940  4.2599959 -1.0245741 -0.6223722  0.4854162 -0.02892339
## [3,] -10.411715 -0.6312698 -1.6526974 -0.5323494  0.6041653 -0.11774348
## [4,]  -8.617296 -2.3229729 -0.2822387 -0.4218828  0.7093994 -0.24722444
## [5,]  -7.542345  1.0853742  0.3561619 -0.1065407 -0.1308295 -0.36834431
## [6,]  -7.565630 -6.0603438  1.8388304 -0.4956560  0.3063986  0.26196247

head(Y)

##           [,1]      [,2]     [,3]      [,4]      [,5]      [,6]
## [1,] -64.87341 -41.84505 25.32417 -15.39198 -30.05441 -1.714282
## [2,] -67.36502 -48.66144 25.47772 -16.03400 -31.33531 -2.216466
## [3,] -66.55825 -43.77018 26.10584 -16.12402 -31.45406 -2.305286
## [4,] -68.35267 -42.07847 24.73539 -16.23449 -31.55929 -2.434767
## [5,] -69.42762 -45.48682 24.09698 -16.54983 -30.71906 -2.555887
## [6,] -69.40433 -38.34110 22.61432 -16.16072 -31.15629 -1.925580

## Here they are really different: prcomp by default centers the data (when we
## did it manually we didn't centere the data). To obtain the same we can just
## center before applying eigenvectors

head(Xc %*% A)              ## ... or

##           [,1]       [,2]       [,3]      [,4]       [,5]        [,6]
## [1,] 12.096558  2.5563952  0.8710214 1.2643930  0.7954799  0.47326072
## [2,]  9.604940 -4.2599959  1.0245741 0.6223722 -0.4854162 -0.02892339
## [3,] 10.411715  0.6312698  1.6526974 0.5323494 -0.6041653 -0.11774348
## [4,]  8.617296  2.3229729  0.2822387 0.4218828 -0.7093994 -0.24722444
## [5,]  7.542345 -1.0853742 -0.3561619 0.1065407  0.1308295 -0.36834431
## [6,]  7.565630  6.0603438 -1.8388304 0.4956560 -0.3063986  0.26196247

head(scale(X, T, F) %*% A) ## ... the same

##           [,1]       [,2]       [,3]      [,4]       [,5]        [,6]
## [1,] 12.096558  2.5563952  0.8710214 1.2643930  0.7954799  0.47326072
## [2,]  9.604940 -4.2599959  1.0245741 0.6223722 -0.4854162 -0.02892339
## [3,] 10.411715  0.6312698  1.6526974 0.5323494 -0.6041653 -0.11774348
## [4,]  8.617296  2.3229729  0.2822387 0.4218828 -0.7093994 -0.24722444
## [5,]  7.542345 -1.0853742 -0.3561619 0.1065407  0.1308295 -0.36834431
## [6,]  7.565630  6.0603438 -1.8388304 0.4956560 -0.3063986  0.26196247
```

- other stuff such as `center` (means), `scale` (logical)

`princomp` `princomp` perform PCA using eigen, as we did by hand. argument of princomp

- the first is the $n \times p$ data matrix
- `cor`, should be set to `TRUE` if the sample correlation matrix is used for PCA and to `FALSE` if the sample covariance matrix is employed.
- if you set `scores=TRUE` (default), then the output will also include an $n \times p$ matrix of principal component scores.

The outcome object is a list of various entries:

- `sdev` gives the standard deviations (the square root of the eigenvalues) of the components
- `loadings` is a matrix with the eigen vectors in the columns (notice that only loadings in modulus larger than 0.1 are printed)
- `center` gives the average vector

```
(princomp_job <- princomp(X, cor = FALSE))

## Call:
## princomp(x = X, cor = FALSE)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 5.4551078 2.7326192 1.9255256 0.5355766 0.3831248 0.3425137
##
##  6  variables and  50 observations.

summary(princomp_job) # same stuff as before

## Importance of components:
##                           Comp.1    Comp.2     Comp.3      Comp.4      Comp.5
## Standard deviation     5.4551078 2.7326192 1.92552560 0.535576640 0.383124752
## Proportion of Variance 0.7173417 0.1800021 0.08937539 0.006914529 0.003538342
## Cumulative Proportion  0.7173417 0.8973438 0.98671916 0.993633685 0.997172027
##                            Comp.6
## Standard deviation     0.342513705
## Proportion of Variance 0.002827973
## Cumulative Proportion  1.000000000

## extract the eigenvalues
## -----------------------
princomp_job$sdev^2  # same as manually

##     Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6
## 29.7582011  7.4672075  3.7076488  0.2868423  0.1467846  0.1173156

spectral$values      # princomp uses biased version of covariance
```

```
## [1] 29.7582011  7.4672075  3.7076488  0.2868423  0.1467846  0.1173156

## matrix of loadings (A)
## ----------------------
princomp_job$loadings # loadings in absolute value < than 0.1 are omitted

##
## Loadings:
##            Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## commun      0.492                0.381  0.756  0.186
## probl_solv        -0.690  0.718
## logical     0.137 -0.705 -0.695
## learn       0.509                0.330 -0.331 -0.718
## physical    0.433                0.218 -0.561  0.670
## appearance  0.534  0.103        -0.836
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000

print(princomp_job$loadings, cutoff = 0)

##
## Loadings:
##            Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## commun      0.492  0.087  0.012  0.381  0.756  0.186
## probl_solv  0.087 -0.690  0.718 -0.009  0.018 -0.007
## logical     0.137 -0.705 -0.695 -0.016  0.016 -0.017
## learn       0.509  0.080  0.021  0.330 -0.331 -0.718
## physical    0.433  0.040  0.010  0.218 -0.561  0.670
## appearance  0.534  0.103  0.022 -0.836  0.067 -0.024
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000

spectral$vectors

##             [,1]        [,2]        [,3]         [,4]        [,5]         [,6]
## [1,] -0.49155807  0.08748571  0.01214618 -0.380639251  0.75573635 -0.185864772
## [2,] -0.08675503 -0.69046582  0.71781729  0.008879732  0.01827673  0.007480667
## [3,] -0.13662756 -0.70496706 -0.69539380  0.015845890  0.01597971  0.016594522
## [4,] -0.50947274  0.08031174  0.02069595 -0.329958740 -0.33064205  0.717887386
## [5,] -0.43261611  0.04023081  0.01036964 -0.217845082 -0.56076816 -0.670223722
## [6,] -0.53428266  0.10274312  0.02196412  0.835735944  0.06699316  0.023681484

## scores Y
## --------

head(princomp_job$scores)
```

```
##          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5      Comp.6
## [1,] -12.096558  2.5563952  0.8710214 -1.2643930  0.7954799 -0.47326072
## [2,]  -9.604940 -4.2599959  1.0245741 -0.6223722 -0.4854162  0.02892339
## [3,] -10.411715  0.6312698  1.6526974 -0.5323494 -0.6041653  0.11774348
## [4,]  -8.617296  2.3229729  0.2822387 -0.4218828 -0.7093994  0.24722444
## [5,]  -7.542345 -1.0853742 -0.3561619 -0.1065407  0.1308295  0.36834431
## [6,]  -7.565630  6.0603438 -1.8388304 -0.4956560 -0.3063986 -0.26196247
```

```
head(Y)          # diff because princomp centers the data
```

```
##           [,1]      [,2]     [,3]      [,4]      [,5]      [,6]
## [1,] -64.87341 -41.84505 25.32417 -15.39198 -30.05441 -1.714282
## [2,] -67.36502 -48.66144 25.47772 -16.03400 -31.33531 -2.216466
## [3,] -66.55825 -43.77018 26.10584 -16.12402 -31.45406 -2.305286
## [4,] -68.35267 -42.07847 24.73539 -16.23449 -31.55929 -2.434767
## [5,] -69.42762 -45.48682 24.09698 -16.54983 -30.71906 -2.555887
## [6,] -69.40433 -38.34110 22.61432 -16.16072 -31.15629 -1.925580
```

```
head(Xc %*% A)   # if we do the same we obtain same results as princomp
```

```
##           [,1]       [,2]       [,3]      [,4]       [,5]        [,6]
## [1,] 12.096558  2.5563952  0.8710214 1.2643930  0.7954799  0.47326072
## [2,]  9.604940 -4.2599959  1.0245741 0.6223722 -0.4854162 -0.02892339
## [3,] 10.411715  0.6312698  1.6526974 0.5323494 -0.6041653 -0.11774348
## [4,]  8.617296  2.3229729  0.2822387 0.4218828 -0.7093994 -0.24722444
## [5,]  7.542345 -1.0853742 -0.3561619 0.1065407  0.1308295 -0.36834431
## [6,]  7.565630  6.0603438 -1.8388304 0.4956560 -0.3063986  0.26196247
```

3.11.1.3 Choosing number of principal components

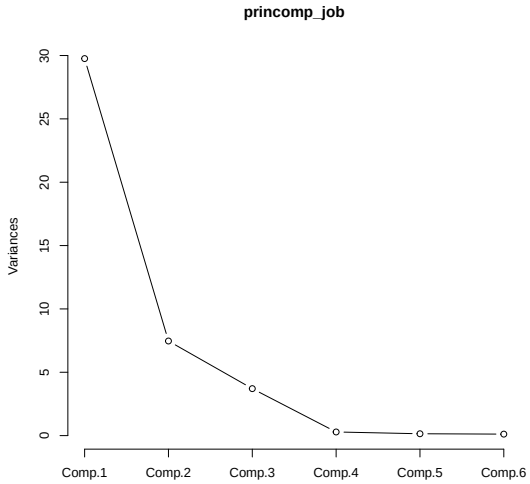
Cumulative proportion of variance explained This approach suggests to choose as many principal components as are needed to explain approximately 80-90% of the total variance. Therefore, in this case the number of PCs needed is 2.

```
cumsum(spectral$value) / sum(spectral$value) # cumulative percentage explained
```

```
## [1] 0.7173417 0.8973438 0.9867192 0.9936337 0.9971720 1.0000000
```

Screeplot The *scree plot* is a plot of l_k against k ($k = 1, \ldots, p$). Usually, the curve tends to become flat after a sharp decline. The flat portion corresponds to noise components, that are not able to capture the leading variability. Therefore, the criterion suggests to retain a number of components equal to the value of k at which the elbow of the scree plot occurs. According to this criterion, the number of PCs to be selected here would be 3 (comp4, 5, 6 are flat).

```
plot(princomp_job, type = "lines")
```




```
spectral$values # last three are close to zero

## [1] 29.7582011  7.4672075  3.7076488  0.2868423  0.1467846  0.1173156
```

Kaiser's rule According to this criterion, the number of principal components to retain is the number of components whose variance is higher than the average variance. Thus, in this case the solution is 2.

```
princomp_job$sdev^2 > mean(princomp_job$sdev^2) # the first two component are selecte

## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
##   TRUE   TRUE  FALSE  FALSE  FALSE  FALSE
```

3.11.1.4 Interpretation of loadings

We choos to work with 2 PC, we can try to interpret it focusing on loadings

```
princomp_job$loadings[, 1:2] # matrix eigenvectors, first two columns

##                Comp.1      Comp.2
## commun     0.49155807  0.08748571
## probl_solv 0.08675503 -0.69046582
## logical    0.13662756 -0.70496706
## learn      0.50947274  0.08031174
## physical   0.43261611  0.04023081
## appearance 0.53428266  0.10274312
```

In the table above:

- first PC (that accounts for about the 72% of the total variability) all the loadings but `probl_solv` and `logical` (which are close to zero) have the

same size and almost the same magnitude. Hence, this component appears to be a sort of average of the measurements.

- In the second PC the only two loadings that are not close to zero are those related to the logical ability and problem solving. Therefore it separates police officers with a marked logical skill (with a small score on the test, as the magnitude is negative) from the others.

Alternatively, it is possible to interpret PCs by evaluating the correlations between each principal component and all the observed variables, i.e:

$$\text{Corr}\,(y_1, \mathbf{x}) = l_1^{1/2}\mathbf{a}_1^T\mathbf{D}^{-1/2}$$
$$\text{Corr}\,(y_2, \mathbf{x}) = l_2^{1/2}\mathbf{a}_2^T\mathbf{D}^{-1/2}$$

where $\mathbf{D}$ is the diagonal matrix containing the original variances.

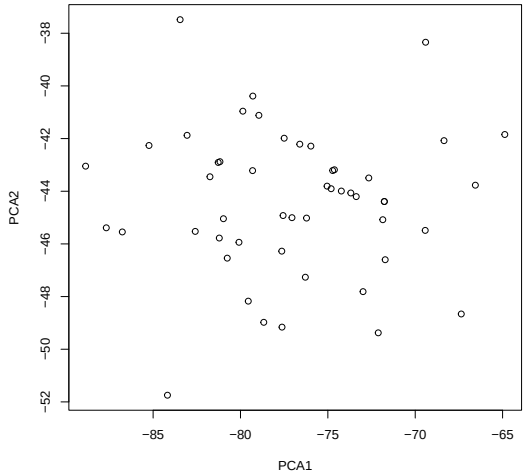
```
## we need to compute D
D <- diag(diag(S))

# Applying the formula above, for the first principal component
# sqrt(l1)                *  a_1^T          D^{-1/2}
sqrt(spectral$values[1]) * t(A[, 1])   %*% solve(D^0.5)

##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] -0.9872352 -0.1983234 -0.3027748 -0.9899587 -0.9889676 -0.9838213

# or just by cor  is the samne
cor(Y[, 1], X)

##          commun probl_solv    logical      learn   physical appearance
## [1,] -0.9872352 -0.1983234 -0.3027748 -0.9899587 -0.9889676 -0.9838213

## similar thing for the second
spectral$values[2]^0.5 * t(A[, 2])   %*% solve(D^0.5)

##            [,1]       [,2]      [,3]       [,4]       [,5]       [,6]
## [1,] 0.08801541 -0.7906737 -0.782575 0.07817195 0.04606954 0.09477061

cor(Y[, 2], X)

##          commun probl_solv   logical      learn   physical appearance
## [1,] 0.08801541 -0.7906737 -0.782575 0.07817195 0.04606954 0.09477061
```

The first principal component is highly negatively related with all the observed variables except for `probl_solv` and logical. Thus, all variables contribute more or less in the same way to the first principal component, confirming the interpretation given above. The second principal component is highly negatively correlated with `l_solv` and `logical`; thus, it separates police officers with a marked logical skill (with a small score on the test, as the magnitude is negative) from the others.

3.11.1.5 Visualization of PC

Since we have just two components we can visualize them. The object scores is a matrix 50×2 containing the scores of the units with respect to the first two principal components.

```
plot(Y[, 1], Y[ , 2], xlab = "PCA1", ylab = "PCA2")
```



3.11.1.6 Are all the units well represented by the bi-dimensional plot?

As the bi-dimensional plot is just an approximation of the structure in the original p-dimensional space, it may happen that not all of the n statistical units are well represented in the bi-dimensional projection.
A way to check the adequacy of the approximation for each statistical unit is an inspection of the scores on the last (discarded) PCs. In particular, if a unit has high scores on the last PCs, this means that it is largely displaced when projected on the PC plane, i.e. it is misrepresented.
One could consider, for each statistical unit, the sum of the squared scores on the last $p - m$ PCs. The position on the PC plane of those units having a large value of such a sum should be carefully considered.

```
(d <- apply(Y[, 3:6], 1, function(x) sum(x^2)))
```

```
##  [1] 1784.433 1893.018 1936.171 1877.315 1804.755 1746.998 1684.826 1743.272
##  [9] 1880.757 1915.520 1880.757 1681.001 1854.007 1668.917 1964.539 1859.880
## [17] 1727.699 1865.102 1759.574 2104.255 1863.209 1873.058 2184.831 1703.492
## [25] 1707.550 1831.796 1938.200 1732.942 1830.631 1912.833 1741.321 1780.767
## [33] 1745.660 1921.742 1795.591 1862.057 1854.136 1722.972 1894.969 1891.101
## [41] 1756.409 1970.561 1901.513 1819.324 1868.010 1812.201 1717.760 1811.706
## [49] 1900.057 1828.348
```

Since these values are high, it is possible to conclude that the first two PCs are not enough to well represent this dataset.
We can find the observations that are represented the worst using the `order` function.

```
head(d_order <- order(d, decreasing=TRUE)) # these are the worst represented

## [1] 23 20 42 15 27  3

job[d_order[1:5], ]

##    commun probl_solv logical learn physical appearance
## 23     18         59      20    50       54         20
## 20     17         57      20    49       54         20
## 42     20         56      22    53       56         24
## 15     16         55      22    49       53         19
## 27     18         57      24    51       54         21
```

3.11.2 Sparrows dataset

Now we do the same for the `sparrow` dataset (but we use correlation instead of covariance matrix). In February 1898, there was a severe winter storm with rain, sleet, and snow near Providence, RI (Rhode Island). 136 English sparrows were found freezing and brought to Dr. Bumpus' laboratory at Brown University. Of those, 72 survived and 64 died. Bumpus took advantage of the opportunity to study an episode of natural selection. He measured a number of characteristics of the birds and analyzed them to find differences between the survivors and those that perished. Here we consider a sample of 49 sparrows. The dataset sparrows.dat contains the following variables:

- `totL`: total length
- `AlarE`: alar extent
- `bhL`: length of beak and head
- `hL`: length of humerus
- `kL`: length of keel of sternum

```
sparrows <- read.table("multivariate_statistics/data/sparrows.dat", header = TRUE)

## Warning in file(file, "rt"):  non è possibile aprire il file 'multivariate_statistics/data/s
File o directory non esistente
## Error in file(file, "rt"):  non è possibile aprire la connessione

head(sparrows)

## Error:  oggetto 'sparrows' non trovato

summary(sparrows)

## Error:  oggetto 'sparrows' non trovato
```

```
(R <- cor(sparrows)) # correlations are high so it makes sense
```

```
## Error:  oggetto 'sparrows' non trovato
```

Is PCA justified Since the correlation values look quite large, in this context PCA seems to be useful in order to reduce dimensionality.

```
# decide to work with covariance or correlation
sapply(sparrows, var)
```

```
## Error:  oggetto 'sparrows' non trovato
```

```
## we have different magnitudes of variances: the first two variables takes
## values that are large, while last three have different units of
## measuremente.  so we standardize data (and use covariance matrix) or use
## directly correlation matrix.
```

Decide between covariance and correlation Moreover, since in this example variances are very different, it is advisable to use the correlation matrix to carry out the PCs. This is equivalent to perform the PCA on the covariance matrix of the standardized data.

Using matrix functions Principal Components can be derived from the spectral decomposition of the correlation matrix by using the `eigen` function.

```
spectral <- eigen(R)
spectral$values # eigevalues
```

```
## [1] 4.03516605 1.26093572 0.63059444 0.03460586 0.02232708 0.01637085
```

```
A <- spectral$vectors # eigevectors
rownames(A) <- rownames(R)
A
```

```
##            [,1]        [,2]        [,3]         [,4]         [,5]          [,6]
## [1,] -0.4898867  0.10661812 -0.01931189 -0.335344756  0.677694372  0.4201724803
## [2,] -0.1111630 -0.72089324 -0.68377657  0.008701665  0.018150128 -0.0004383121
## [3,] -0.1636455 -0.66488745  0.72831261  0.015284637  0.019952761 -0.0085914040
## [4,] -0.4913118  0.09769660 -0.02568178 -0.262692420  0.009760144 -0.8242001751
## [5,] -0.4920224  0.06977629 -0.01668897 -0.266769473 -0.733753782  0.3784259095
## [6,] -0.4872378  0.11161019 -0.02644446  0.864327077  0.038896332  0.0294785507
```

To compute the PCA scores, in order to obtain the same result as `princomp`, x should be scaled

```
head(scores <- as.matrix(sparrows) %*% A)
```

```
## Error:  oggetto 'sparrows' non trovato
```

```
head(scores_stand <- scale(sparrows, T, T) %*% A) # to get same results as princomp we scale
```

```
## Error:  oggetto 'sparrows' non trovato
```

Compute the importance of each component in explaining the total variance

```
# proportion explained
spectral$values / sum(spectral$values) # prop explained
```

```
## [1] 0.672527675 0.210155954 0.105099073 0.005767644 0.003721180 0.002728475
```

```
cumsum(spectral$values) / sum(spectral$values) # first two components are enough
```

```
## [1] 0.6725277 0.8826836 0.9877827 0.9935503 0.9972715 1.0000000
```

```
# total variability: is number of variables working with std data the varainces
# is 1)
sum(spectral$values)
```

```
## [1] 6
```

```
sum(diag(R))
```

```
## [1] 6
```

We find that the first PC accounts for the 72.3% of the total variation, whereas the first two components cover about the 83% of the total variability. Notice that the sum of the eigenvalues is equal to the number of variables as the variance of each standardized variable is 1.

Using `princomp` on correlation matrix Here we use `cor=TRUE`, so princpomp know it has to standaridze the data

```
pca_s <- princomp(sparrows, cor = TRUE, scores=TRUE)
```

```
## Error:  oggetto 'sparrows' non trovato
```

```
## eigenvalues
## -----------
pca_s$sdev^2    # are the
```

```
## Error:  oggetto 'pca_s' non trovato
```

```
spectral$values #  same
```

```
## [1] 4.03516605 1.26093572 0.63059444 0.03460586 0.02232708 0.01637085
```

```
## eigenvectors
## -----------
print(pca_s$loadings, cutoff = 0)   # same up to sign
```

```
## Error:  oggetto 'pca_s' non trovato
```

```
spectral$vectors # change (see the uper matrix)
```

```
##            [,1]        [,2]        [,3]         [,4]         [,5]          [,6]
## [1,] -0.4898867  0.10661812 -0.01931189 -0.335344756  0.677694372  0.4201724803
## [2,] -0.1111630 -0.72089324 -0.68377657  0.008701665  0.018150128 -0.0004383121
## [3,] -0.1636455 -0.66488745  0.72831261  0.015284637  0.019952761 -0.0085914040
## [4,] -0.4913118  0.09769660 -0.02568178 -0.262692420  0.009760144 -0.8242001751
## [5,] -0.4920224  0.06977629 -0.01668897 -0.266769473 -0.733753782  0.3784259095
## [6,] -0.4872378  0.11161019 -0.02644446  0.864327077  0.038896332  0.0294785507
```

```
## prop variance explained
## -----------
summary(pca_s)
```

```
## Error:  oggetto 'pca_s' non trovato
```

Regarding $\mathbf{Y}$ there is a slight difference between the two results; this is due to the fact that scale uses unbiased variances to standardize, while `princomp` does not!

```
## Y
## --
head(pca_s$scores) # princomp computes the scores after centering and scaling x
```

```
## Error:  oggetto 'pca_s' non trovato
```

```
head(scores) # without standardizing data they're vastly different
```

```
## Error:  oggetto 'scores' non trovato
```

```
# with standardization: equal to pca_s$scores up to a sign change!
head(scores_stand) # standardizing using unbiased covmatrix
```

```
## Error:  oggetto 'scores_stand' non trovato
```

The final standardized score are actually similar but not equivalent to those returned by `princomp` stored in `pca_s` (some decimals are changed): the reason is that `scale` use the unbiased covariance matrix, while `princomp` uses the biased ones.
To get the exactly what with `princomp` does:

```
n <- nrow(sparrows)
```

```
## Error:  oggetto 'sparrows' non trovato
```

```
C <- diag(n) - 1/n * rep(1, n) %*% t(rep(1, n)) # centering matrix
Xc <- C %*% as.matrix(sparrows)  #centered data
```

```
## Error:  oggetto 'sparrows' non trovato
```

```
S <- 1/n * t(Xc) %*% Xc
D <- diag(diag(S))
Z <- Xc %*% solve(D^0.5)
```

```
scores_z <- Z %*% A
scores_z[1:5, ] # now exactly equivalent up to sign changes to

##          [,1]       [,2]       [,3]       [,4]        [,5]         [,6]
## [1,] 4.593657  0.9260852 -0.3843684 0.46840774  0.33965785 -0.047771308
## [2,] 3.440966 -1.8416263 -0.3608873 0.20053062 -0.19328148 -0.042553546
## [3,] 3.846632  0.1450697 -0.6847236 0.16475441 -0.24852154 -0.028041971
## [4,] 3.213781  0.8902318 -0.1438149 0.12039210 -0.30511080  0.001972478
## [5,] 2.740818 -0.4850129  0.1635882 0.02285044 -0.01607894  0.145827499

pca_s$scores[1:5, ]

## Error:  oggetto 'pca_s' non trovato
```

Select n of components According to this criterion we should retain as many principal components as are needed to explain approxi- mately 80-90% of the total variance. Thus, in this case the number of PC needed is 2.

```
summary(pca_s) # two principal components

## Error:  oggetto 'pca_s' non trovato
```

According to this criterion, the number of PCs to be selected here would be 2.

```
plot(pca_s, type="l") # screeplot here seems that two components are enough

## Error:  oggetto 'pca_s' non trovato
```

The Kaiser's rule for PCs derived from a correlation matrix suggests to retain only those components that exhibit eigenvalues of R larger than 1. The motivation underlying this rule is that we want to retain all the principal components that have a variance larger than the one related to the original variables (that is equal to 1 for standardized data). In this case only the first Principal Component should be used.

```
pca_s$sdev^2 >mean(pca_s$sdev^2) # just the first

## Error:  oggetto 'pca_s' non trovato
```

Interpretation of loadings Taking the first two components ...

```
# --------------
# interpretation
# --------------
pca_s$loadings[ , 1:2]

## Error:  oggetto 'pca_s' non trovato
```

In the first PC (that accounts for about the 72% of the total variability) all the loadings have the same size and almost the same magnitude. Hence, this component appears to be a sort of average size of the sparrows. Even if the second PC is not important in recovering a reduced dimension representation of our data, we can try to give it an empirical meaning. Specifically, It appears to be a linear contrast as some variables have a positive coefficient and some others have a negative one. It separates sparrows having a small alar extent and short humerus, a small beak and a large keel of sternum from all the others. Therefore, it can be thought of as a measure of shape. The first loading is close to zero so it is not interpretable. The same result can be obtained by analyzing the correlations between each PC and all the observed variables:

```
## first component
spectral$values[1]^0.5  *  t(A[, 1]) %*% solve(diag(diag(R))^0.5)

##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] -0.9840708 -0.2233011 -0.3287266 -0.9869336 -0.9883609 -0.9787497

cor(scores_stand[, 1], X) # same result

## Error:  oggetto 'scores_stand' non trovato

## second component
spectral$values[2]^0.5  *  t(A[, 2]) %*% solve(diag(diag(R))^0.5)

##          [,1]       [,2]       [,3]      [,4]       [,5]      [,6]
## [1,] 0.119723 -0.8095011 -0.7466114 0.1097049 0.07835277 0.1253286

cor(scores_stand[, 2], X) # same result

## Error:  oggetto 'scores_stand' non trovato
```

The first principal component is highly negatively correlated with each of the observed variables. All variables contribute more or less in the same way to the first principal component, confirming the interpretation given above. The second principal component is negatively correlated with AlarE, bhL, hL and positively related with kL, separating sparrows having a small alar extent and short humerus, a small beak and a large keel of sternum from all the others.

Plotting of first two components The object scores is a matrix 49×2 containing the scores of the units with respect to the first two principal components.

```
plot(scores_stand[, 1], scores_stand[, 2])

## Error:  oggetto 'scores_stand' non trovato
```

Are all the units well represented by the bi-dimensional plot? Notice that the sligth difference below are due to the reason explained before, i.e. `scale` uses corrected variances to standardize, while princomp not!

```
apply(pca_s$scores[, -(1:2)], 1, function(x) sum(x^2))     # biased std
```

```
## Error:  oggetto 'pca_s' non trovato
```

```
(d <- apply(scores_stand[, -(1:2)], 1, function(x) {sum(x^2)})) # correct std
```

```
## Error:  oggetto 'scores_stand' non trovato
```

```
sort(d, decreasing =TRUE)
```

```
##  [1] 2184.831 2104.255 1970.561 1964.539 1938.200 1936.171 1921.742 1915.520
##  [9] 1912.833 1901.513 1900.057 1894.969 1893.018 1891.101 1880.757 1880.757
## [17] 1877.315 1873.058 1868.010 1865.102 1863.209 1862.057 1859.880 1854.136
## [25] 1854.007 1831.796 1830.631 1828.348 1819.324 1812.201 1811.706 1804.755
## [33] 1795.591 1784.433 1780.767 1759.574 1756.409 1746.998 1745.660 1743.272
## [41] 1741.321 1732.942 1727.699 1722.972 1717.760 1707.550 1703.492 1684.826
## [49] 1681.001 1668.917
```

Since these values are low, it is possible to conclude that the first two PCs are enough to well represent this dataset.

Capitolo 4

Factor analysis

4.1 Introduction

Remark 50. Factor analysis is a statistical model that allows to explain the correlations between a large number of observed correlated variables through a small number of uncorrelated unobservable ones: the factors.

Remark 51 (History). This method was invented by Spearman (psychometrician) in 1904 to measure intelligence (Galton wanted to study correlation of intelligence between father and sons; but measure of intelligence was not available so he measured height, something measurable). Spearman's work provided a very clever and useful tool that is still at the bases of the most advanced instruments for measuring intelligence.
Intelligence is the prototype of a vast class of variables that are not directly observed (they are called latent variables) but can be measured in an indirect way through the analysis of observable variables closely linked to the latent ones. Latent variables are common to many research fields besides psychology, from medicine to genetics, form finance to economics and this explain the still vivid interest towards Factor analysis.

4.1.1 Simple linear factor model

Example 4.1.1. Spearman had a dataset on students performance in three subjectss' test: x_1 classics, x_2 french and x_3 english with correlation matrix

$$\boldsymbol{\rho} = \begin{bmatrix} 1 & 0.83 & 0.78 \\ & 1 & 0.67 \\ & & 1 \end{bmatrix}$$

Variables are positive correlated (with high correlation). Some question:

- is this correlation due to another variable which drives the correlation (he called it general ability/intelligence) ?
- if so if I condition/eliminate on the unobserved/latent variable, does the observed correlation diminish? in this case the correlation is due to the hidden variable (it's the same of partial correlation but with a variable which is not observed)

Remark 52. Again for every topic the notation changes (λ below has nothing to do with lambda of PCA)

Important remark 42 (Sperman's model). So is there a latent variable that allows to explain the observed correlation? Spearman assumed that the performance (random variables on single students) is composed as

$$x_1 = \lambda_1 f + u_1$$
$$x_2 = \lambda_2 f + u_2$$
$$x_3 = \lambda_3 f + u_3$$

where:

- x_1, x_2 and x_3 are the *observed variables* (grades);
- the quantity f, with no subscript, is the same for all the variables and called *common factor*;
- λ_1, λ_2 and λ_3 are known as *factor loadings*: these indicate how much the common factor contributes to the different observed values of the variables
- the $u_1, \ldots, u_3$ are different for each variable and are known as *unique/specific factors*. The unique factors represent residuals, random noise terms that besides telling that an examination only offers an approximate measure of the subject's ability, also describe, for each individual, how much his result on a given subject, differs from his general ability.

The idea that Spearman had was (considering performance in classics x_1 for example):

- I have a given level of intelligence f (which is a random variable)
- my result in classics (x_1) is due to my intelligence, weighed by a contribution λ_1 which is common/fixed (not related to the single unit)
- then there is u_1, a random variable (like f) which takes into account all other factors impacting performance in classics other than intelligence. This is like ε term in regression: it's random variability around the model.

Remark 53. After posing the model, Spearman research opened big debate: people started to put question regarding intelligence: is there a unique intelligence? Actually no, there are different side of intelligence, which is a good motivating example for introducing the the general model of intelligence.

4.1.2 General linear factor model

Remark 54. Spearman's model can be generalized to include more than one common factor

Important remark 43 (General linear factor model). A generic test x_i can be explained in terms of m intelligence components as

$$x_i = \lambda_{i1} f_1 + \ldots + \lambda_{ik} f_k + \ldots \lambda_{im} f_m + u_i$$

where

- there are many f for different kind of intelligence;
- the weights/λ has *two subscript*: the first concerning the *observed/explained variable*, the second the *latent*/intelligence one)

Remark 55. This factor model can be written in matrix form as follows: we are considering *a single unit* (with p observed variables to be explained and m latent one), *not a sample of units*.

Important remark 44 (Model in matrix form). Let $\mathbf{x}$ be a p-dimensional random vector with expected value μ and covariance matrix $\mathbf{\Sigma}$. An m factor model for $\mathbf{x}$ holds if it can be decomposed as:

$$\underset{p\times 1}{\mathbf{x}} = \underset{p\times m}{\mathbf{\Lambda}}\ \underset{m\times 1}{\mathbf{f}} + \underset{p\times 1}{\mathbf{u}} + \mu$$

If we assume to deal with mean centered $\mathbf{x}$ variables then, with no loss in generality, the model will be:

$$\underset{p\times 1}{\mathbf{x}} = \underset{p\times m}{\mathbf{\Lambda}}\ \underset{m\times 1}{\mathbf{f}} + \underset{p\times 1}{\mathbf{u}} \tag{4.1}$$

where:

- $\mathbf{\Lambda}$ is the *factor loading matrix* ($p \times m$ because of p observed variables and m latent ones); explicitly it is

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_{11} & \dots & \lambda_{1k} & \dots & \lambda_{1m} \\ \dots & & & & \\ \lambda_{i1} & \dots & \lambda_{ik} & \dots & \lambda_{im} \\ \dots & & & & \\ \lambda_{p1} & \dots & \lambda_{pk} & \dots & \lambda_{pm} \end{bmatrix}$$

  What does factor *loadings* mean? For example in the first row we have the contribution of each common factor to variable x_1. It tells how much each factor "loads" to obtain the observed variable; it might be that not all the factor are relevant for a certain observed variable.

- $\mathbf{f} = \begin{bmatrix} f_1 & \dots & f_k & \dots & f_m \end{bmatrix}^\top$ is the random vector of *common factors*;
- $\mathbf{u} = \begin{bmatrix} u_1 & \dots & u_i & \dots & u_p \end{bmatrix}^\top$ is the random vector of *unique factors*.

Important remark 45 (Comparison with other stats models). We have that:

- factor analysis ($\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u}$) is a linear model aimed at explaining *observed* correlation (between observed variables)
- $\mathbf{u}$ has the same role of residual ε in linear models (difference first related to observed variable, second to units);
- if data have been generated as above I can explain the correlation between observed variables $(x_1, \dots, x_3)$, as we'll see;
- peculiarity here is that all what is on the rhs of the model is unknown/unobserved (different from linear models where ε is the only unknown/random);

- as in PCA, this model perform *dimension reduction*: observed variables are p while latent factors are m which is expected to be smaller than p;
- two appreciable differences with respect to PCA:
  - what Spearman did was to figure out *a model* which could explain the observed correlation: any model require *assumptions* (while in PCA we had no theoretical assumptions);
  - the focus here is to explain *correlation* (off diagonal elements of variance/covariance/correlation matrix), while in PCA was on variance (in diagonal elements)

Remark 56 (Estimability and assumptions). As said all what appears on the right hand side of equal sign is unknown. In order to be able to estimate the model we need to make assumptions and thus to put constraints on $\mathbf{f}$ and $\mathbf{u}$.[1]

4.2 Assumptions

Remark 57. We have the following assumptions: the first twos are just simplifying while the last two are the most important.

Important remark 46. Assumptions of the linear factor models:

1. we work with $\mathbf{0}$ centered random variables:

$$\mathbb{E}[\mathbf{f}] = \mathbf{0}, \quad \mathbb{E}[\mathbf{u}] = \mathbf{0}$$

   This is just a low-cost simplifying assumption, which can be easily obtained by working with mean centered observations.[2] This first assumption eases the estimation of a simpler model without loss of generality.

2. variance covariance matrix of latent variables is identity, so we assume that the common factors have unit variance and are uncorrelated:

$$\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right] = \mathrm{Var}[\mathbf{f}] = \mathbf{I}_m$$

   with the first equivalence due to $\mathbf{f}$ being mean centered.
   This again is a simplifying assumptions: it's not fundamental (we will relax it) and it can be proved (using Choleski decomposition) that any model can be transformed to a model respecting this condition;

[1] When we estimate a model, to estimate we always make assumptions: eg in regression ε has 0 mean, constant variance and uncorrelated between obs. We need similar conditions here, but on $\mathbf{f}$ and $\mathbf{u}$ (while in regression was only on ε)

[2] If I had not been working with mean center obs I had

$$x_i = \mu_i + \lambda_{i1} f_1 + \ldots + \lambda_{im} f_m + u_i$$

but μ_i is a complication that adds nothing since we're interested in correlations. If we work with mean centered data, to have that $\mathbb{E}[x_i] = \mu_i = 0$, we have

$$\underbrace{\mathbb{E}[x_i]}_{=0} = 0 + \lambda_1 \underbrace{\mathbb{E}[f_1]}_{=0} + \ldots + \lambda_{im} \underbrace{\mathbb{E}[f_m]}_{=0} + \underbrace{\mathbb{E}[u]}_{=0}$$

and thus the assumption made ($\mathbb{E}[\mathbf{f}] = \mathbf{0}$, $\mathbb{E}[\mathbf{u}] = \mathbf{0}$. O quanto meno questo è il meglio che ho tirato giù.)

3. the first fundamental/characterizing assumption is that **common factors and the unique factors are uncorrelated**, that is $\text{Cov}(\mathbf{f}, \mathbf{u}) = \text{Cov}(\mathbf{u}, \mathbf{f}) = \mathbf{0}$ (eg there's no correlation between intelligence and other aspect/noise that impact on performance of a score). In matrix form the assumption can be expressed as:

$$\mathbb{E}\left[\mathbf{f}\mathbf{u}^\top\right] = \underset{m \times p}{\mathbf{0}}$$
$$\mathbb{E}\left[\mathbf{u}\mathbf{f}^\top\right] = \underset{p \times m}{\mathbf{0}}$$

these are the variance/covariance between $\mathbf{f}$ and $\mathbf{u}$ (null matrices of different sizes).

4. variance-covariance of unique factors $\mathbf{u}$ is diagonal so unique factors are uncorrelated among them (eg other things influencing my performance on classics are not correlated with other things influencing my performance in math) and may be heteroscedastic. In matrix form:

$$\mathbb{E}\left[\mathbf{u}\mathbf{u}^\top\right] = \text{Var}\left[\mathbf{u}\right] = \mathbf{\Psi}$$

with $\mathbf{\Psi}$ a diagonal matrix such as

$$\mathbf{\Psi} = \begin{bmatrix} \psi_{11} & \dots & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & \psi_{ii} & \dots & 0 \\ \dots & & & & \\ 0 & \dots & 0 & \dots & \psi_{pp} \end{bmatrix}$$

The variances ψ_{ii} are called **uniquenesses** or **specific variances**: so we have specific/different variances for each factor

4.3 Implications

4.3.1 Observed variance and covariance

Now assuming that all those conditions are met, we're able to compute the covariance between the observed variables as:

$$\begin{aligned} \mathbf{\Sigma} \overset{(0)}{=} \mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right] &= \mathbb{E}\left[(\mathbf{\Lambda}\mathbf{f} + \mathbf{u})(\mathbf{\Lambda}\mathbf{f} + \mathbf{u})^\top\right] \\ &= \mathbb{E}\left[\mathbf{\Lambda}\mathbf{f}\mathbf{f}^\top\mathbf{\Lambda}^\top + \mathbf{\Lambda}\mathbf{f}\mathbf{u}^\top + \mathbf{u}\mathbf{f}^\top\mathbf{\Lambda}^\top + \mathbf{u}\mathbf{u}^\top\right] \\ &\overset{(1)}{=} \mathbf{\Lambda}\,\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right]\mathbf{\Lambda}^\top + \mathbf{\Lambda}\,\mathbb{E}\left[\mathbf{f}\mathbf{u}^\top\right] + \mathbb{E}\left[\mathbf{u}\mathbf{f}^\top\right]\mathbf{\Lambda}^\top + \mathbb{E}\left[\mathbf{u}\mathbf{u}^\top\right] \\ &\overset{(2)}{=} \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi} \end{aligned} \tag{4.2}$$

where in

- (0) the equality holds because still working with mean centered data;
- (1) we split the expected value; $\mathbf{\Lambda}$ are constants to be put outside expectation;

- (2) we substituted $\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right] = \mathbf{I}$ (assumption 2), $\mathbb{E}\left[\mathbf{f}\mathbf{u}^\top\right] = \mathbf{0}$ and $\mathbb{E}\left[\mathbf{u}\mathbf{f}^\top\right] = \mathbf{0}$ (assumption 3) and $\mathbb{E}\left[\mathbf{u}\mathbf{u}^\top\right] = \mathbf{\Psi}$ (assumption 4);

Important remark 47. Thus:

- if the factor model holds the covariance matrix of observed variables can be decomposed as

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$$

  Just for info the opposite is also true: if the covariance matrix of the observed variables $\mathbf{x}$ can be decomposed as in equation 4.2 then the linear factor model 4.1 holds;

- being $\mathbf{\Psi}$ diagonal, this means that the observed covariances (off diagonal elements in $\mathbf{\Sigma}$) are completely determined/accounted for/by the common factors via $\mathbf{\Lambda}\mathbf{\Lambda}^\top$

So:

- if the data have been generated according to model $\mathbf{x} = \mathbf{\Lambda}\mathbf{f} + \mathbf{u}$
- if our assumptions on $\mathbf{f}$ and $\mathbf{u}$ holds

we can write $\mathbf{\Sigma}$ as linear combination of $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ and we can explain correlation using only factors (since the remaining $\mathbf{\Psi}$ is diagonal). It's very important that $\mathbf{\Psi}$ is diagonal to say that there's no residual correlation.
To expand a bit the matrix notation, indeed:

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$$

$$\begin{bmatrix} \sigma_1^2 & \dots & \dots & \dots & \sigma_{1p} \\ & \dots & \dots & \dots & \dots \\ & & \sigma_i^2 & \dots & \sigma_{ip} \\ & & & \dots & \dots \\ & & & & \sigma_p^2 \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \dots & \lambda_{1k} & \dots & \lambda_{1n} \\ \dots & & & & \\ \lambda_{i1} & \dots & \lambda_{ik} & \dots & \lambda_{in} \\ \dots & & & & \\ \lambda_{p1} & \dots & \lambda_{pk} & \dots & \lambda_{pn} \end{bmatrix} \begin{bmatrix} \lambda_{11} & \dots & \lambda_{i1} & \dots & \lambda_{p1} \\ \dots & & & & \\ \lambda_{1k} & \dots & \lambda_{ik} & \dots & \lambda_{pk} \\ \dots & & & & \\ \lambda_{1n} & \dots & \lambda_{in} & \dots & \lambda_{pn} \end{bmatrix}$$

$$+ \begin{bmatrix} \psi_{11} & \dots & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & \psi_{ii} & \dots & 0 \\ \dots & & & & \\ 0 & \dots & 0 & \dots & \psi_{pp} \end{bmatrix}$$

4.3.1.1 Single variances decomposition

Focusing on generic element σ_i^2 on diagonal of $\mathbf{\Sigma}$, it's generated considering the i-th row of $\mathbf{\Lambda}$, i-th column of $\mathbf{\Lambda}^\top$ (that is i-th row of $\mathbf{\Lambda}$ again) and the ii-th element of Ψ:

$$\sigma_i^2 = \underbrace{\sum_{k=1}^{m} \lambda_{ik}^2}_{h_i^2} + \psi_{ii} = \underbrace{\lambda_{i1}^2 + \ldots + \lambda_{ik}^2 + \ldots + \lambda_{in}^2}_{h_i^2} + \psi_{ii}$$

so if the factor model holds, the **observed variance of a variable can be decomposed** in

- a first part due to the components/lambda variables (*common part*). The following

$$h_i^2 = \sum_{k=1}^{m} \lambda_{ik}^2$$

  is called *communality*: it's the part of the observed variance that is accounted for/by the common factor; or, in other words, it is the variance of x_i that is shared with the other variables via the common factors.

- a second unique part due to ψ_{ii}, called *the uniqueness* (or unique variance): it is the variance of the i-th unique factor and represents the part of the variance of x_i not accounted for by the common factors.

So we can say that the observed variance can be decomposed in the sum of communality + uniqueness. The latent variables are able to explain just a part of the observed variability; then there's the specific part off course.

4.3.1.2 Covariances splitting

Focusing on the off diagonal elements of $\mathbf{\Sigma}$, say a generic covariace σ_{ip}, this latter is obtained multiplying the i-th row of $\mathbf{\Lambda}$ and p-th column of $\mathbf{\Lambda}^\top$ (last row of $\mathbf{\Lambda}$), thus

$$\sigma_{ip} = \sum_{k=1}^{m} \lambda_{ik}\lambda_{pk} + 0$$

(at the end added 0 because off diagonal element of $\mathbf{\Psi}$, $\psi_{ip} = 0$).
So here if the model holds, all the observed covariance are fully explained by the latent ones.

4.3.1.3 Variance explained by latent factors

Important remark 48 (Variance of variable explained by a single factor). Considering again the definition we've given on the communality, the part of variance of x_i which is explained by the m common factor:

$$h_i^2 = \sum_{k=1}^{m} \lambda_{ik}^2 = \lambda_{i1}^2 + \lambda_{i2}^2 + \ldots + \lambda_{im}^2$$

If we focus on a single terms of the sum we have that eg λ_{i1}^2 is the part of variance of x_i which is explained by the first common factor.

Important remark 49 (Total variance explained by a single factor). If I consider the trace of covariance matrix $\operatorname{Tr}\mathbf{\Sigma}$ I have the total variance (in case of standardized variables the total variance is equal to the number of variables ($\operatorname{Tr}\boldsymbol{\rho} = p$). If I sum with respect to all the variables (that is column sum of squares of $\mathbf{\Lambda}$)

$$\sum_{i=1}^{p} \frac{\lambda_{i1}^2}{p}$$

i obtain the *part of total variance explained by factor 1.*
So I have an idea of how a factor explain.

4.3.2 Covariance between observed variables and latent factors (Λ)

Let's derive the covariance between the observed variables $\mathbf{x}$ and the latent factors $\mathbf{f}$, which being mean/zero centered is the following expected value:

$$\text{Cov}(\mathbf{x}, \mathbf{f}) = \mathbb{E}\left[\mathbf{x}\mathbf{f}^\top\right]$$

If the model holds, $\mathbf{x} = \mathbf{\Lambda}\mathbf{f} + \mathbf{u}$, then:

$$\mathbb{E}\left[\mathbf{x}\mathbf{f}^\top\right] = \mathbb{E}\left[(\mathbf{\Lambda}\mathbf{f} + \mathbf{u})\mathbf{f}^\top\right] = \mathbb{E}\left[\mathbf{\Lambda}\mathbf{f}\mathbf{f}^\top + \mathbf{u}\mathbf{f}^\top\right] \overset{(1)}{=} \mathbf{\Lambda}\underbrace{\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right]}_{\mathbf{I}} + \underbrace{\mathbb{E}\left[\mathbf{u}\mathbf{f}^\top\right]}_{\mathbf{0}} \overset{(2)}{=} \mathbf{\Lambda}$$

where in

- (1) we take out $\mathbf{\Lambda}$ (which acts as constant)
- (2) we substitute from assumptions $\mathbb{E}\left[\mathbf{u}\mathbf{f}^\top\right] = \mathbf{I}$ and $\mathbb{E}\left[\mathbf{u}\mathbf{f}^\top\right] = \mathbf{0}$

Important remark 50. So beside being the factor loading matrix, $\mathbf{\Lambda}$ is *also* the covariance matrix between observed variable $\mathbf{x}$ and latent/common factor $\mathbf{f}$.

4.3.3 Scale equivariance

Remark 58. Consider the following scale change transformation of $\mathbf{x}$

$$\mathbf{y} = \mathbf{C}\mathbf{x}$$

where $\mathbf{C}$ is a *diagonal, square* scale change matrix: by pre-multiplying by $\mathbf{C}$ we obtain a new vector $\mathbf{y}$ where the i-th component is the corresponding i-th of $\mathbf{x}$ multiplied by the ii-th element of $\mathbf{C}$.

Important remark 51. It turns out that the linear factor model is *equivariant* with respect to scale changes. If we substitute the factor model for the original variable:

$$\mathbf{y} = \mathbf{C}\mathbf{x} = \mathbf{C}(\mathbf{\Lambda}_\mathbf{x}\mathbf{f} + \mathbf{u}_\mathbf{x}) = \underbrace{\mathbf{C}\mathbf{\Lambda}_\mathbf{x}}_{\mathbf{\Lambda}_\mathbf{y}}\mathbf{f} + \underbrace{\mathbf{C}\mathbf{u}_\mathbf{x}}_{\mathbf{u}_\mathbf{y}} = \mathbf{\Lambda}_\mathbf{y}\mathbf{f} + \mathbf{C}\mathbf{u}_\mathbf{y}$$

we obtain a new factor model (for $\mathbf{y}$) where:

- the common factor $\mathbf{f}$ is the same;
- the new factor loading matrix $\mathbf{\Lambda}_\mathbf{y}$ is just a rescaled version of the old one applying $\mathbf{C}$, that is $\mathbf{\Lambda}_\mathbf{y} = \mathbf{C}\Lambda_\mathbf{x}$;
- the new unique variance is actually

  $$\mathbf{\Psi}_\mathbf{y} = \text{Var}\left[\mathbf{u}_\mathbf{y}\right] = \mathbb{E}\left[\mathbf{u}_\mathbf{y}\mathbf{u}_\mathbf{y}^\top\right] = \mathbb{E}\left[\mathbf{C}\mathbf{u}_\mathbf{x}\mathbf{u}_\mathbf{x}^\top\mathbf{C}\right] = \mathbf{C}\mathbf{\Psi}_\mathbf{x}\mathbf{C}$$

  indeed, developing variance decomposition of the transformed variables:

  $$\begin{aligned}\text{Var}\left[\mathbf{y}\right] = \text{Var}\left[\mathbf{C}\mathbf{x}\right] &\overset{(0)}{=} \mathbf{C}\,\text{Var}\left[\mathbf{x}\right]\mathbf{C}^\top \overset{(1)}{=} \mathbf{C}\mathbf{\Sigma}\mathbf{C} = \mathbf{C}(\mathbf{\Lambda}_\mathbf{x}\mathbf{\Lambda}_\mathbf{x}^\top + \mathbf{\Psi}_\mathbf{x})\mathbf{C} \\ &= \underbrace{\mathbf{C}\mathbf{\Lambda}_\mathbf{x}}_{\mathbf{\Lambda}_\mathbf{y}}\underbrace{\mathbf{\Lambda}_\mathbf{x}^\top\mathbf{C}}_{\mathbf{\Lambda}_\mathbf{y}^\top} + \underbrace{\mathbf{C}\mathbf{\Psi}_\mathbf{x}\mathbf{C}}_{\mathbf{\Psi}_\mathbf{y}}\end{aligned}$$

  where (0) is the vector equivalent of $\text{Var}\left[aX\right] = a^2\,\text{Var}\left[X\right]$ and in (1) we removed transposition from $\mathbf{C}$ since it's symmetric.

Remark 59. This proves that linear factor model is scale equivariant. Implication is that if we rescale *we don't need to refit* the model but just multiply the factor loadings for the matrix $\mathbf{C}$.
The scale equivariance property implies that we can define the model either on the raw or on the standardized data and it is always possible to go from one solution to the other by applying the same scale change to the model parameters. It is worth reminding that, on the contrary, PCs don't share this property.

Example 4.3.1 (Standardization)**.** A common re-scaling transformation is standardization, that amounts to choose as transformation matrix $\mathbf{C} = \mathbf{\Delta}^{-1/2}$ (considering $\mathbf{\Delta}$ the diagonal matrix with variances of the $\mathbf{x}$ variables, $\mathbf{\Delta}^{-1/2}$ has diagonal elements which are $1/\sqrt{\sigma^2_{ii}}$), and then standardizing with

$$\mathbf{y} = \mathbf{\Delta}^{-1/2}\mathbf{x}$$

$\mathbf{y}$ are standardized variables (zero mean and unitary variance/SD); the corresponding factor loadings matrix $\mathbf{\Lambda_y}$ will be:

$$\mathbf{\Lambda_y} = \mathbf{\Delta}^{-1/2}\mathbf{\Lambda_x}$$

So the factor loading matrix referred to the standardized variables is equal to the factor loading matrix of the raw data multiplied by $\mathbf{\Delta}^{-1/2}$.
So interestingly

- we can fit/model on the original data and go to the results/model/solution with standardized data by just pre-applying $\mathbf{\Delta}^{-1/2}$
- viceversa apply $\mathbf{\Delta}^{1/2}$ to go from standardized results to the unstandardized ones.

This is not the case with PCA.
Finally what is $\mathbf{\Lambda_y}$ when standardizing?

$$\mathbf{\Lambda_y} = \mathbf{\Delta}^{-1/2}\mathbf{\Lambda_x} = \mathbf{\Delta}^{-1/2}\operatorname{Cov}(\mathbf{x},\mathbf{f}) = \mathbf{\Delta}^{-1/2}\operatorname{Cov}(\mathbf{x},\mathbf{f})\underbrace{\mathbf{I}^{-1/2}}_{\text{sd of } \mathbf{f}}$$

so $\mathbf{\Lambda_y}$ is the correlation matrix between $\mathbf{x}$ and $\mathbf{f}$, $\mathbf{\Lambda_y} = \operatorname{Corr}(\mathbf{x},\mathbf{f})$.
So if we work on original data $\mathbf{\Lambda_x}$ is the *covariance* matrix between the observed and the latent variables; if otherwise we work on the standardized data the $\mathbf{\Lambda_y}$ is the *correlation* matrix between the observed and the latent variables.

4.4 Model identifiability

Before moving to estimation issues we need to focus a little bit on what is known as identifiability.

Definition 4.4.1 (Identifiability)**.** A statistical model is said to be *identifiable* if a solution to the estimation problem *exists* and *is unique*; or in other words if different values of the parameters estimated generate different predicted values.

Important remark 52. Actually our our linear factor model is not identifiable:

- condition imposed so fare are not enough to prove that there is just one solution: there are infinity of factor models satisfying all the conditions above and thus a single factor model is not identifiable; this is due to the fact that linear factor model is *invariant after orthogonal rotations.*
- in other words there is an infinite number of different matrices $\mathbf{\Lambda}$ that can generate the same $\mathbf{x}$ values: this is due to the fact that linear factor model is *invariant after orthogonal rotations.*

So anyone trying to estimate a model can obtain a different legitimate estimate.

Important remark 53. The invariance after orthogonal rotations seems a drawback but it will be useful in the future. Once we fix this identifiability problem it turns out to be a benefit.

Remark 60. Below we prove that identifiability does not hold looking at existence and uniqueness of solutions.

Uniqueness Let's consider a $m \times m$ orthogonal matrix called $\mathbf{G}$, such that

$$\mathbf{G}^\top \mathbf{G} = \mathbf{G}\mathbf{G}^\top = \mathbf{I}$$

now our model can be rewritten as

$$\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u} \overset{(1)}{=} \mathbf{\Lambda I f} + \mathbf{u} \overset{(2)}{=} \underbrace{\mathbf{\Lambda G}}_{\mathbf{\Lambda}^*} \underbrace{\mathbf{G}^\top \mathbf{f}}_{\mathbf{f}^*} + \mathbf{u}$$

where

- in (1) we inserted an $\mathbf{I}$
- in (2) we just replaced by $\mathbf{G}\mathbf{G}^\top$

Calling $\mathbf{\Lambda G}$ as $\mathbf{\Lambda}^*$ and $\mathbf{G}^\top \mathbf{f}$ as $\mathbf{f}^*$ we end up with

$$\mathbf{x} = \mathbf{\Lambda}^* \mathbf{f}^* + \mathbf{u}$$

So we can take infinite number of orthogonal matrices and we obtain two models that has the same properties. Let's prove that the two models:

$$\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u}$$
$$\mathbf{x} = \mathbf{\Lambda}^* \mathbf{f}^* + \mathbf{u}$$

are completely equivalent. They have the same properties since:

$$\mathbb{E}[\mathbf{f}^*] = \mathbb{E}\left[\mathbf{G}^\top \mathbf{f}\right] = \mathbf{G}^\top \cdot \underbrace{\mathbb{E}[\mathbf{f}]}_{=\mathbf{0}} = \mathbf{0}$$

$$\mathbb{E}\left[\mathbf{f}^* \mathbf{f}^{*\top}\right] = \mathbb{E}\left[\mathbf{G}^\top \mathbf{f}\mathbf{f}^\top \mathbf{G}\right] = \mathbf{G}^\top \cdot \underbrace{\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right]}_{\mathbf{I}} \mathbf{G} = \mathbf{G}^\top \mathbf{G} = \mathbf{I}$$

$$\mathbb{E}\left[\mathbf{f}^* \mathbf{u}^\top\right] = \mathbb{E}\left[\mathbf{G}^\top \mathbf{f}\mathbf{u}^\top\right] = \mathbf{G}^\top \underbrace{\mathbb{E}\left[\mathbf{f}\mathbf{u}^\top\right]}_{=\mathbf{0}} = \mathbf{0}$$

so the new model has latent factor with null expected values, have unit variances and are uncorrelate; finally common factor and unique factor are uncorrelated; thus the two models have the same property/are indistinguishable.

Important remark 54. So actually we have an infinity (as the orthogonal matrices are) of equivalent solutions: for this reason people started to try to find an unique solution. In order to guarantee that a unique solution is obtained we need to fix something else. Constraints on $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ are usually imposed; the most common ones are that either $\mathbf{\Lambda}^\top \mathbf{\Psi} \mathbf{\Lambda}$ or $\mathbf{\Lambda}^\top \mathbf{\Delta}^{-1} \mathbf{\Lambda}$ are diagonal.

Important remark 55. We further impose the constraint that:

$$\mathbf{\Lambda}^\top \mathbf{\Psi} \mathbf{\Lambda} \quad \text{is diagonal} \tag{4.3}$$

this is a constraint based on Bayesian inference (basically all the software do this): $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ cannot be any but have to be so that the quadratic above is diagonal. By putting this constraint we obtain a single solutions.

Remark 61. When we'll move to interpretation having a lot of solution due to factor rotation will be useful.

Remark 62. Now we proved how to obtain a unique solution, we need to verify if it exists.

Existence As said all what appears on right hand side of $\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u}$ is unknown; so there's a lot of things to be estimated.
It's not possible to estimate all in one shot: first we estimate $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ (for $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$) and then we estimate $\mathbf{f}$. Also this problem of estimating $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ is a little bit tricky. The starting point of our estimation procedure

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$$

where we know the elements of $\mathbf{\Sigma}$ only.
How many are the *unique* elements of $\mathbf{\Sigma}$, our pieces of information? They are variances and unique covariances so it's the sum of first p natural numbers, that is $p(p+1)/2$.
We want to reconstruct these elements using $p(p+1)/2$ equations and find the unknowns on the other side: our unknowns are the $p \cdot m$ elements of $\mathbf{\Lambda}$ plus p diagonal elements of $\mathbf{\Psi}$, so $p \cdot m + p$ unknowns.
In order to obtain of solutions the number of equations must be larger than the number of unknowns; to reduce the number of unknows we impose the constraint in eq 4.3, that is $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ must satisfy $\mathbf{\Lambda}^\top \mathbf{\Psi} \mathbf{\Lambda}$ is diagonal. This reduces the number of unknowns since off diagonal elements fixed to be zero: the matrix is $m \times m$ so we have $m \cdot (m-1)/2$ element fixed/constrained to 0.
By applying the constraint we have:

$$\frac{p(p+1)}{2} \quad \text{equations}$$
$$pm + p - \frac{m(m-1)}{2} \quad \text{unknowns}$$

If the first number is lower than the second we have an infinite number of solutions; if equality holds then we have a unique exact solution that however is useless from a statistical point of view since it doesn't allow to explain the observed correlations through a small number of latent variables.
Thus in order for a solution to exists it must be:

$$\frac{p(p+1)}{2} > pm + p - \frac{m(m-1)}{2} \tag{4.4}$$

where:

- p (number of variables we have) is given
- we need to chose m (maximum number of common factors/latent variables) such that this equality is satisfied (so not all the value of m).

Remark 63. Equation 4.4, known in literature as *Lederman's condition*, is the condition of maximum number of common factor we can include in a factor model with p observed variable.

Remark 64. In order to have solution one could set $\geq$ in the Lederman's inequality: the reason we put strict $>$ is that we want to have dimension reduction as well.

4.5 Model estimation

In practice we do not know $\mathbf{\Sigma}$ (the population covariance matrix), but can estimate it by the sample covariance matrix $\mathbf{S}$ (using either the correct or biased version); or if we want to deal with standardized data we estimate population correlation $\boldsymbol{\rho}$ by the sample counterpart $\mathbf{R}$.
Once done that estimation is performed in two stages:

1. we try to estimate population's $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ to obtain this decomposition

   $$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$$

   the estimation/decomposition is performed on sample quantities:

   $$\mathbf{S} = \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top + \hat{\mathbf{\Psi}}$$

   how to decompose $\mathbf{S}$ in the sum above? This estimation issue can be tackled in at least **two different ways**/approaches.[3]

2. then we estimate the factor scores $\mathbf{f}$

For the first step we see

1. *principal factor method*: it is a distribution free method, we don't need to make any assumption on the probability density function of $\mathbf{x}$ in the population. It's one of the older heuristic methods still often used;
2. the best known *maximum likelihood method*, that assumes normality, $\mathbf{x}$ is normally distributed

Both methods need to start from a *preliminary solution*; we start from a plausible estimate of the communalities and refine it until we find a good enough solution.

[3]Various estimation methods have been proposed over the years. Most of them lacked in theoretical bases, and that's one reason why factor analysis didn't have much success among the statistical community and remained confined to the psychological literature. Only around 1960 Joreskog succeeded in deriving a sound and reliable algorithm for maximum likelihood estimation. This greatly helped the diffusion of factor analysis that, to day, is one of the most widely used dimension reduction models.

To recall, the communalities are the p quantities (one for each observed variable x_i):

$$h_i^2 = \sum_{k=1}^{m} \lambda_{ik}^2$$

that is

- communality of x_i is the part of the variance of the observed x_i explained by the common factors (shared by all the variables);
- for each row of $\mathbf{\Lambda}$ the communality is the sum of its squares.

Communalities kickstarting As starting point for communalities we can adopt different strategies, depending on we're working with correlation $\mathbf{R}$ (standardized variables) or covariance $\mathbf{S}$ (unstandardized variables):

1. R^2**-like measures**: to have the part explained we can take the R_{i0}^2 of x_i regressed on all the *other/remaining* $x_{i'}$[4]. In case of:
   - $\mathbf{R}$: we use it directly, that is
   $$\hat{h}_i^2 = R_{i0}^2$$
   - $\mathbf{S}$: we multiply it by the i-th variable variance to have the quantity explained
   $$\hat{h}_i^2 = s_i^2 \cdot R_{i0}^2$$
2. use the **max (absolute) correlation** among x_i and any other variable $x_{i'}$ (absolute value is needed because percentage explained is positive). With
   - $\mathbf{R}$: we use it directly
   $$\hat{h}_i^2 = \max_{i'} |r_{ii'}|$$
   - $\mathbf{S}$: we again multiply it by the i-th variable variance
   $$\hat{h}_i^2 = s_i^2 \cdot \max_{i'} |r_{ii'}|$$
3. set to the **maximum value possible**: by doing this *we obtain principal component*. For the interpretation it means that *unique factors does not exists/aren't needed*: common factors are able to explain variances/covariances. For:
   - $\mathbf{R}$
   $$\hat{h}_i^2 = 1$$
   - $\mathbf{S}$:
   $$\hat{h}_i^2 = s_i^2$$

[4]It measures the portion of the variance of x_i that is explained by its linear relationship with the other variables. As the communality is the part of the variance of x_i that is shared with the other variables via the common factors, R_{i0}^2 can provide a reasonable approximation.

4.5.1 Principal factor method

Let's start from $\mathbf{S}$: we build what we call the *reduced covariance matrix*. At population level we have that

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi} \implies \mathbf{\Sigma} - \mathbf{\Psi} = \mathbf{\Lambda}\mathbf{\Lambda}^\top$$

In the sample its equivalent is

$$\mathbf{S} = \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top + \hat{\mathbf{\Psi}} \implies \mathbf{S} - \hat{\mathbf{\Psi}} = \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$$

with $\mathbf{S} - \hat{\mathbf{\Psi}}$ called *reduced covariance matrix* ($\mathbf{R} - \mathbf{\Psi}$ is called *reduced correlation matrix*).The *reduced covariance matrix* is:

- almost identical to $\mathbf{S}$, but with communality on main diagonal instead of variances:

$$\mathbf{S} - \hat{\mathbf{\Psi}} = \begin{bmatrix} \hat{h}_1^2 & s_{12} & \dots & s_{1p} \\ & \hat{h}_2^2 & \dots & s_{2p} \\ \dots & & & \\ & & & h_p^2 \end{bmatrix}$$

- is still squared symmetric (changed only the diagonal of a symmetric matrix) but *no longer positive semi-definite* (we've changed diagonal entries[5]): this means that no longer eigenvalue will be ≥ 0.

As squared symmetric, reduced matrix can be decomposed using spectral decomposition as:

$$\mathbf{S} - \hat{\mathbf{\Psi}} = \mathbf{\Gamma}\mathbf{L}\mathbf{\Gamma}^\top$$

where $\mathbf{\Gamma}$ is the orthonormal matrix whose columns are the eigenvectors of $\mathbf{S}$ and $\mathbf{L}$ is the diagonal matrix of the eigenvalues.
Now if we consider the *positive eigenvalue only*, supposing the first m are, a rank m approximation of $\mathbf{S} - \hat{\mathbf{\Psi}}$ can thus be obtained as

$$\mathbf{S} - \hat{\mathbf{\Psi}} \overset{(0)}{\approxeq} \mathbf{\Gamma}_m\mathbf{L}_m\mathbf{\Gamma}_m^\top \overset{(1)}{=} \underbrace{\mathbf{\Gamma}_m\mathbf{L}_m^{1/2}}_{\hat{\mathbf{\Lambda}}}\underbrace{\mathbf{L}_m^{1/2}\mathbf{\Gamma}_m}_{\hat{\mathbf{\Lambda}}^\top}$$

where in

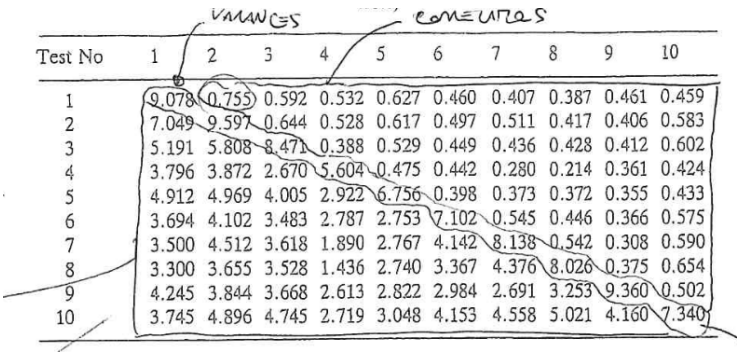
- (0) $\mathbf{L}_m$ is the diagonal matrix of the positive eigenvalues (and $\mathbf{\Gamma}_m$ the corresponding eigenvectors);
- (1) we substituted $\mathbf{L}_m = \mathbf{L}_m^{1/2}\mathbf{L}_m^{1/2}$

So using the spectral theorem I found a way to have an estimate of

$$\hat{\mathbf{\Lambda}} = \mathbf{\Gamma}_m\mathbf{L}_m^{1/2}$$

Once estimated $\hat{\mathbf{\Lambda}}$, the diagonal elements of $\hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$ provide new estimates of the communalities. The estimation procedure can be stopped here or iterated, by replacing the new communality estimates on the diagonal of the reduced covariance matrix, until convergence.

[5] When we proved that S is positive semi-definite we used variances and covariances and the relationships that exists between variances and covariances. Now we've destroyed the main diagonal changing it and losing positive-semidefiniteness.



Test No	1	2	3	4	5	6	7	8	9	10
1	9.078	0.755	0.592	0.532	0.627	0.460	0.407	0.387	0.461	0.459
2	7.049	9.597	0.644	0.528	0.617	0.497	0.511	0.417	0.406	0.583
3	5.191	5.808	8.471	0.388	0.529	0.449	0.436	0.428	0.412	0.602
4	3.796	3.872	2.670	5.604	0.475	0.442	0.280	0.214	0.361	0.424
5	4.912	4.969	4.005	2.922	6.756	0.398	0.373	0.372	0.355	0.433
6	3.694	4.102	3.483	2.787	2.753	7.102	0.545	0.446	0.366	0.575
7	3.500	4.512	3.618	1.890	2.767	4.142	8.138	0.542	0.308	0.590
8	3.300	3.655	3.528	1.436	2.740	3.367	4.376	8.026	0.375	0.654
9	4.245	3.844	3.668	2.613	2.822	2.984	2.691	3.253	9.360	0.502
10	3.745	4.896	4.745	2.719	3.048	4.153	4.558	5.021	4.160	7.340

Figura 4.1: Children example: variance and correlations

Important remark 56 (Scale equivariance). The factor model is scale equivariant; however here we're estimating factor model using spectral decomposition which is not scale equivariant. Thus

- *estimates obtained by the principal factor method are not scale equivariant*
- we cannot transform estimate obtained with unstandardized variable to obtain the estimate obtained with standardized variables.

As a consequence, model fitted on the raw data or on the standardized ones are different and there is no way to go from one solution to the other. Most statistical software perform FA on $\mathbf{R}$ as a default option

Remark 65. Finally to obtain the estimate $\hat{\mathbf{\Psi}}$, we just apply:

$$\hat{\mathbf{\Psi}} = \mathbf{S} - \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$$

Important remark 57 (Residual covariance matrix). This last matrix

$$\hat{\mathbf{\Psi}} = \mathbf{S} - \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$$

or the standardized equivalent residual correlation matrix

$$\hat{\mathbf{\Psi}} = \mathbf{R} - \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$$

is know as the *residual covariance matrix* (respectively correlation) and constitutes an *important heuristic tool to check model fit* of the m factor solution.
We expected $\mathbf{\Psi}$ to be diagonal, if it's (almost) diagonal we're happy; if otherwise large entries are present in the off-diagonal part of $\hat{\mathbf{\Psi}}$, it turns out that the common factors are unable to explain fully the observed covariances/correlations and our model gives poor fit to the data. If $\hat{\mathbf{\Psi}}$ elements out of main diagonal are small it's still OK.
Poor performance could be due:

- take into account too few factors;
- relationship between variable is not linear (our model is linear)

Example 4.5.1 (Esempio cartaceo scores bambini)**.** A data collection on children intelligence in 10 test:

Manifest variable		Factor loadings		Communalities	
		Factor 1	Factor 2	Initial	Final
1	(Information)	0.776	−0.333	0.659	0.713
2	(Vocabulary)	0.823	−0.224	0.689	0.728
3	(Arithmetic)	0.731	−0.055	0.527	0.537
4	(Similarities)	0.589	−0.248	0.403	0.419
5	(Comprehension)	0.678	−0.239	0.478	0.517
6	(Animal house)	0.668	0.148	0.451	0.468
7	(Picture completion)	0.647	0.293	0.469	0.505
8	(Mazes)	0.627	0.379	0.492	0.535
9	(Geometric design)	0.562	0.022	0.331	0.316
10	(Block designs)	0.789	0.321	0.664	0.726

Figura 4.2: Children example: factor loadings and communalities

1. information
2. vocabulary
3. arithmetic
4. similarities
5. comprehension
6. animal house
7. picture completion
8. mazes
9. geometric design
10. block design

the first five seems to be math/linguistic related, the last actorial/visual space. A matrix of correlation (upper triangular) and variances covariances (lower triangular) is given in 4.1. The following analysis was performed on correlation matrix. In the example there are some pretty high correlation: pupils with high grades on some subject tends to have high grades in other subjects.
In figure 4.2 the factor loading matrix $\hat{\mathbf{\Lambda}}$ (first two columns) and communality estimates for a two factor model fitted (using principal factor method on the correlation matrix) so the model is something like

$$x_1 = \lambda_{11} f_1 + \lambda_{12} f_2$$
$$\dots$$
$$x_{10} = \lambda_{10,1} f_1 + \lambda_{10,2} f_2$$

for the interpretation

1. the first factor is more or less equal important in all the variables (high coefficients and thus correlated with all of them): this factor can be seen as a general value of general intelligence (if high, being positively correlated, one tend to performs well in every subject; while low values of this factor tend to be associated with poor performaces). This is generally considered as general intelligence factor;

2. the second factor: having both positive and negative coefficients can be seen as a contrast (some variables are not relevant here such as geometric design, being their coefficient close to 0). Large values on this factor has very good score on first animal house to block design variables, but lowest scores in first 5 variables. This is math/linguistic vs art/picture tendency.

The second part of table gives info on communalities: in the first columns initial guesses, while in the last column that is the final values are reported:

- the first/initial was calculated using R^2_{i0}, so for example the first 0.659 is the R^2 of information predicted using all the other variables;
- the final communalities/column can be obtained as $\hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$ so for example considering the first one for information

$$\begin{aligned}\hat{h}^2_1 &= \hat{\lambda}^2_{11} + \hat{\lambda}^2_{12}\\ 0.713 &= 0.776^2 + (-0.333)^2\end{aligned}$$

Regarding **explained variance** (looking at the diagonal of variance/covariance or correlation matrix) if we were working on the raw data we have the following decomposition (from $\mathbf{S}' = \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top + \hat{\mathbf{\Psi}}$):

$$\sigma^2_i = \sum_{k=1}^{m} \lambda^2_{ik} + \psi_{ii}$$

In this case, working with standardized data (analysis performed on correlation matrix) we have that $\sigma^2_i = 1$. Anyway:

- $\hat{\lambda}^2_{11}$ is the part of the variance of the first variable X_1 accounted for by f_1. For example the part of variance for first variable (information) explained by the first factor is 0.776^2 (out of 1, being working on standardized variables);
- $\hat{\lambda}^2_{12}$ is the part of variance of X_1 accounted/explained by f_2. For example second variable (vocabulary), first factor's explained variance is 0.823^2
- the communality $\hat{h_i}^2 = \lambda^2_{11} + \lambda^2_{12}$ is the part of the variance of X_1 which is explained by the common factors. In the example the part of variance of information explained the two factors is $0.713 = (0.776)^2 + (-0.333)^2$ (out of overall 1, working with standardized information, so 71%)
- the overall variance is in general $\operatorname{Tr}\mathbf{S}$, in this case $\operatorname{Tr}\mathbf{R} = p$ (that is the number of variables. The percentage of the total variance which is explained by the first factor f_1.

$$\frac{\sum_{i=1}^{p} \hat{\lambda^2_{i1}}}{\operatorname{Tr}\mathbf{S}} \cdot 100$$

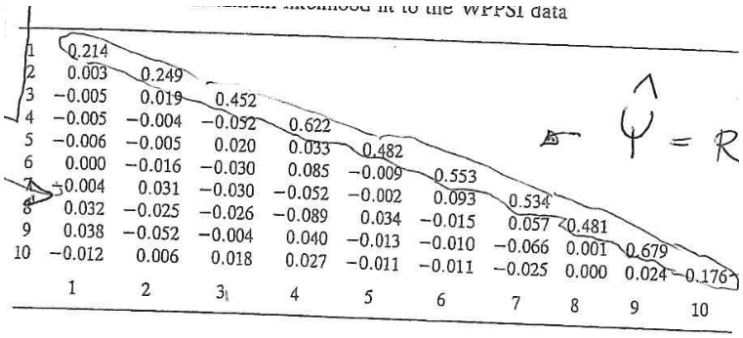


Figura 4.3: Children example: residuals (from ML method)

in the example the first factor explains 48% of total variability, being

$$\frac{0.776^2 + 0.823^2 + \ldots + 0.789^2}{10} \cdot 100 = 48.1\%$$

Now looking out of the main diagonal of variance/covariance or correlation matrix, **how good is my model** in reproducing the observed correlation (or variances/covariances)? In general we have that the observed covariance/correlation, out of corresponding matrix main diagonal, can be decomposed as

$$\sigma_{ii'} = \sum_{k=1}^{m} \lambda_{ik}\lambda_{i'k}$$

(here with standardized data $\sigma_{ii'}$ is a correlation instead of variance).
For example, if we want to study the observed correlation between information and vocabulary which is Corr (Information, Vocabulary) $= 0.755$ (the true correlation in the table), how good is my model in reconstructing it? The correlation between X_1 (information) and X_2 (vocabulary) explained by the common factors is

$$\hat{r}_{12} = \text{Corr}\,(\hat{\text{Information}}, \text{Vocabulary}) = 0.716 \cdot 0.823 + (-0.333) \cdot (-0.224) = 0.71324$$

we hope that $\hat{r}_{12}$ is near to r_{12} observed (0.755). So if we take the difference between observed and reconstructed

$$0.755 - 0.71324 = 0.04176$$

This final one is the entry in the residual correlation of information and vocabulary: given a matrix with all these residuals I can study it to know how well my model fit the data.
In table 4.3 the matrix with these residual[6] values or in other terms we have $\hat{\mathbf{\Psi}} = \mathbf{R} - \hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top$, that is the difference between observed and explained correlations (out of diagonal) we hope they're small, while on the main diagonal we have unique variances ψ_{ii}. This is an empirical way to check model fit, and it's the only way with principal factors method. Actually all the out of diagonal elements are very close to 0; the model gives a pretty good fit to the data.

[6]Residuals are not exactly the same because estimated using ML, while we used principal factors method above.

Important remark 58 (Recap). We're interesting in fitting $\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u}$: this is a linear model which is aimed at explaining covariances (or correlations)
However there are too many parameters (all right hand side is unknown).
Under some assumptions we're able to decompose, if the model holds, $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$$

where $\mathbf{\Lambda}$ is the factor loading matrix, but also it is the covariance matrix between the observed variables and the common factors (if we deal with std variable will be also the correlation matrix between obs value and common factors).
In order to estimate this decomposition $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$ we estimate $\mathbf{\Sigma}$ with $\mathbf{S}$, then we obtain a preliminary estimate for the communalities, which are the diagonal entries of $\mathbf{\Sigma}$

$$\sigma_i^2 = h_i^2 + \psi_{ii}$$

obtaining the reduced covariance matrix $(\mathbf{S} - \hat{\mathbf{\Psi}})$.
There are two methods to estimate the model:

1. *principal factor methods*: based on the spectral decomposition of $\mathbf{S} - \hat{\boldsymbol{Psi}}$ (which is the reduced covariance matrix, $\mathbf{S}$ with diagonal entries replaced with the preliminary communalities). Thus we obtain factor loading matrix etc etc.

2. ML we'll see after

4.5.2 ML method

Remark 66. When Spearman invented Factor analysis, the only estimation available was the previous one (ML was not available, while spectral decomposition was).
Then people prefer to have a way to formally test if say two factor are enough; thus when ML was introduced people reset the problem above with ML.

We need to specify the *distribution in the population*: typically we assume that $\mathbf{x}$ is MVN with parameters $\boldsymbol{\mu}, \mathbf{\Sigma}$:

$$\mathbf{x} \sim \text{MVN}\left(\boldsymbol{\mu}, \mathbf{\Sigma}\right)$$

The pdf of MVN is

$$f(\mathbf{x}) = \det\left(2\pi\mathbf{\Sigma}\right)^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

where within the exponential we have the Mahalanobis distance between a unit and the mean of the distribution, with negative sign so the higher the distance the lower the density.
The likelihood will be (assuming independence of observations):

$$L(\mathbf{x}; \boldsymbol{\mu}, \mathbf{\Sigma}) = \prod_{j=1}^{n} f(\mathbf{x}) = \det\left(2\pi\mathbf{\Sigma}\right)^{-n/2} \exp\left[-\frac{1}{2}\sum_{j=1}^{n}(\mathbf{x}_j-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}_j-\boldsymbol{\mu})\right]$$

Manifest variable		Factor loadings		Communalities	
		Factor 1	Factor 2	Initial	Final
1	(Information)	0.789	−0.403	0.659	0.786
2	(Vocabulary)	0.834	−0.234	0.689	0.751
3	(Arithmetic)	0.740	−0.034	0.527	0.548
4	(Similarities)	0.586	−0.185	0.403	0.378
5	(Comprehension)	0.676	−0.248	0.478	0.518
6	(Animal house)	0.654	0.140	0.451	0.447
7	(Picture completion)	0.641	0.234	0.469	0.466
8	(Mazes)	0.629	0.351	0.492	0.519
9	(Geometric design)	0.564	0.054	0.331	0.321
10	(Block designs)	0.808	0.414	0.664	0.824

Figura 4.4: Children example: ML estimates

The logarithm allows us to get rid of exponential part:

$$l(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{n}{2}\log\det(2\pi\boldsymbol{\Sigma}) - \frac{1}{2}\left[\sum_{j=1}^{n}(\mathbf{x}_j-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}_j-\boldsymbol{\mu})\right]$$

What happens is that we can rewrite this quantity as:

$$l(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{n}{2}\log\det(2\pi\boldsymbol{\Sigma}) - \frac{n}{2}\operatorname{Tr}\boldsymbol{\Sigma}^{-1}\mathbf{S} - \frac{n}{2}(\overline{\boldsymbol{x}}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\overline{\boldsymbol{x}}-\boldsymbol{\mu})$$

The ML estimate for mean $\boldsymbol{\mu}$ is sample mean $\bar{\mathbf{x}}$: if we substitute $\overline{\boldsymbol{x}}$ to $\boldsymbol{\mu}$ then the quadratics in the third term disappears and we have that

$$l(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{n}{2}\log\det(2\pi\boldsymbol{\Sigma}) - \frac{n}{2}\operatorname{Tr}\boldsymbol{\Sigma}^{-1}\mathbf{S}$$

If $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}$ holds, then

$$l(\mathbf{x};\boldsymbol{\Sigma}) = l(\mathbf{x};\boldsymbol{\Lambda},\boldsymbol{\Psi}) = -\frac{n}{2}\log\det\left(2\pi(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top+\boldsymbol{\Psi})\right) - \frac{n}{2}\operatorname{Tr}\left[(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top+\boldsymbol{\Psi})^{-1}\mathbf{S}\right]$$

Deriving with respect to $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ and set the derivatives $= 0$, unfortunately this equation don't admit a closed forme/solution/formula to compute $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ directly.
It took up to the 70's Joreskog provided a numeric algorithm to solve the problem, so numeric methods are used to obtain numeric estimates for $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$.[7]

Example 4.5.2. Back to the sheet example in table 4.4 the estimates of factor loadings (and final communalities) using ML are close but not the same as those in principal factors (tab 4.2). Other than that interpretation of all quantities is the same.

[7] In 1967 Joreskog developed an efficient two-stage algorithm. Convergence is quite fast, but it might happen that one or more elements of the estimate of $\boldsymbol{\Psi}$ become negative. This situation, known as Heywood case, can be solved by constraining the elements of $\boldsymbol{\Psi}$ to be non negative.

4.5.2.1 Scale equivariance

Important remark 59. Differently from previous estimation method (spectral decomposition), one important aspect of ML is that *method* and *estimates* are scale equivariant: if we fit the factor model with ML on unstandardized data, we can go to factor model with standardized data applying the scale change (no need to refit) and viceversa.

Important remark 60. If $\hat{\mathbf{\Lambda}}_{\mathbf{x}}$ be the covariance matrix between the observed variables and the common factors and I am interested in $\hat{\mathbf{\Lambda}}_{\mathbf{z}}$ (referred to the standardized variables) I simply standardize the solution by dividing it by std deviations (at left we have std of $\mathbf{x}$, at right the std of common factor which is standardized with unit variance and zero covariance):

$$\hat{\mathbf{\Lambda}}_{\mathbf{z}} = \mathbf{D}^{-1/2}\hat{\mathbf{\Lambda}}_{\mathbf{x}}\mathbf{I}_m^{-1/2} = \mathbf{D}^{-1/2}\hat{\mathbf{\Lambda}}_{\mathbf{x}}$$

Remark 67. This transformation can be done only with ML estimates because scale equivariant.

4.5.2.2 Hypothesis testing

Before the decision on model was based on residual covariance matrix. Fitting the common factor model by maximum likelihood also allows to select, through a suitable likelihood ratio test, the number of common factors to be included in the model. Under normality assumption we can test if a given number of factors is enough.
The test hypotheses are:

$$H_0 : \mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$$
$$H_1 : \mathbf{\Sigma} \text{ is unstructured (different from model in } H_0)$$

the test statistics is a LRT statistic with formula

$$W = \underbrace{\left[n - \frac{2p+11}{6} - \frac{2m}{3}\right]}_{\text{Correction factor}} \left[\log\det\left(\hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^\top + \hat{\mathbf{\Psi}}\right) - \log\det\left(\mathbf{S}\right)\right]$$

where

- $\hat{\mathbf{\Lambda}}$ and $\hat{\mathbf{\Psi}}$ are the maximum likelihood estimates of the corresponding model parameters;
- between first square brackets we have the Bartlett's correction factor to speed up convergence to χ^2.
- as all LRT is distributed asymptotically as Chisq and precisely:

  $$W \sim \chi^2, \quad \text{with } \frac{1}{2}\left[(p-m)^2 - (p+m)\right] \text{ degrees of freedom}$$

  with df which depend on n of variable and factors (p is fixed, m is change as we proceed in sequential tests).

The testing procedure is **sequential**: we start with a model with 1 factor. is it enough? We estimate all the components in W:

- if we reject the null we need to increment by 1 the number of factors;
- if we don't reject we end with the current number of factors.

Important remark 61. The maximum number of factor one can test is the one imposed by the Lederman's condition. If it occurs that we reject the model with maximum number of factor there's some problem with the model, not the number of factors.
We're looking at a linear model, which turns out to be not appropriate to explain observed covariances: there are other kind of factor model, which are non-linear, and that can do better in explaining data (relationship between variable are better described by a non-linear relationship).

Example 4.5.3. The testing procedure applied to the example yields:

1. fitting a single factor model we have that $W = 58.83$, df $= 35$ and $p = 0.007$: so we decide to reject null (that 1 factor is enough) and need to increment number of factors;
2. with a two factor model: $W = 16.51$, df $= 26$ and $p = 0.92$ so we don't reject the null and the model is OK to take two factor

4.6 Factor rotation

Remark 68. In order to improve interpretability of the factor loadings we can rely on the invariance to orthogonal rotation property of the factor model.

Remark 69. In 1947 Thurstone gave a definition of how an *interpretable/simple factor structure* should be. The variables should be divisible into groups such that the loadings within each group are high on a single factor, perhaps moderate to low on a few factors and negligible on the remaining factors. Eg if we have three common factor with these loadings (H = high, M = moderate, 0 = null), this should be the ideal interpretation situation:

$$\Lambda = \begin{bmatrix} H & M & 0 \\ H & M & 0 \\ H & M & 0 \\ 0 & H & M \\ 0 & H & M \\ M & O & H \\ M & O & H \end{bmatrix}$$

Interpreting here is easier. If the factor structure is simple, the variables should be divisible into groups such that the loadings, within each group, are high on a single factor, perhaps moderate to low on a few factors and negligible on the remaining factor.
However this situation is typically not the case with unrotated factors, where the first is a general factor (and other are contrasts).

Important remark 62. To improve interpretability of the factor loadings we can rely on the invariance to orthogonal rotation property and practically speaking we rotate the solution by post-multiplying the obtained one for an orthogonal matrix.

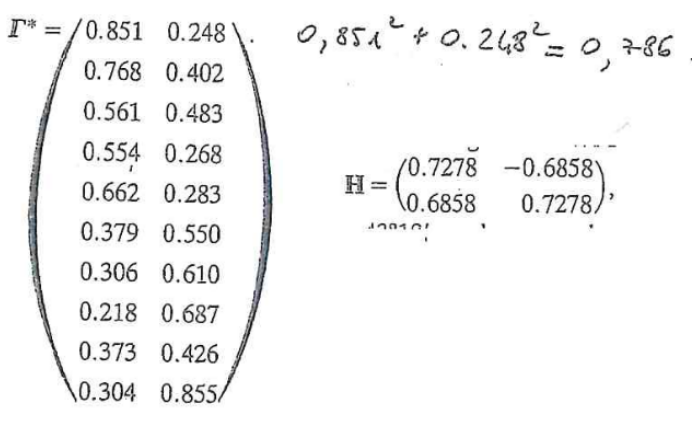


Figura 4.5: Children example: factor rotation

Example 4.6.1. Matrix **H** in the example (table 4.5) is an orthogonal matrix; if we perform an orthogonal rotation of the solution, by computing $\hat{\mathbf{\Lambda}}\mathbf{H}$ we obtain a new solution in $\mathbf{\Gamma}^*$.
The new model has the same properties of the original one: looking back to section , there the orthogonal matrix which here is **H** there was called **G**.
One way to check that the solution is the same is to compute

- the communalities (explained variances): if one do the computations, we obtain the same communalities. Eg for the first variable

$$0.851^2 + 0.248^2 = 0.786$$

- same holds for the explained covariances.

This is what invariance after orthogonal rotation means.

Important remark 63. There are a lot of rotations: most widely known are

- varimax (orthogonal/uncorrelated factors): see it below

- quartimax (orthogonal/uncorrelated factors): maximizes the overall variance of the factor loading matrix, thus usually producing, as the first factor, a general factor with positive and almost equal loadings on all the variables;

- oblimin (not orthogonal, correlated): differently from the previous one it's based on a minimization and produces an *oblique* rotations so it provides *correlated* factors (which can be useful/needed sometimes)

4.6.1 Varimax

Varimax searches for an orthogonal rotation of the factor loading matrix, such that the following criterion is maximized

$$V = \sum_{k=1}^{m} \overbrace{\left[\underbrace{\frac{\sum_{i=1}^{p} \beta_{ik}^4}{p}}_{(A)} - \underbrace{\left(\frac{\sum_{i=1}^{p} \beta_{ik}^2}{p} \right)^2}_{B} \right]}^{C}$$

where

$$\beta_{ik} = \frac{\lambda_{ik}}{\sqrt{\sum_{k=1}^{m} \lambda_{ik}^2}} = \frac{\lambda_{ik}}{h_i}$$

Starting from β_{ik}:

- numerator is usual loading of factor k on variable i;
- denominator is sum of squared loadings in the i-th row with sqrt (sqrt of communality), it's the norm of the row vector
- their ratio is a kind of normalization of λ_{ik} by the row norm
- if we put all the betas in a matrix (call it *normalized matrix*) this matrix has row that have unit norms (so are somewhat comparable between them)

Going to the main equation, we can see inner square bracket summations work inside columns of the obtained normalized matrix: within square brackets there's a sort of variance (one can see the fast variance formula $\mathbb{E}\left[X^2\right] - (\mathbb{E}\left[X\right])^2$) of β_{ik}^2, that is the variance of the betas squared in the considered k-th column:

- (A) is the square of mean of squared β (the $\mathbb{E}\left[X^2\right]$ component)
- (B) is the mean of squared β (kind of variance), squared (the $(\mathbb{E}\left[X\right])^2$ component of variance)
- (C) is variance of squared β values in column k (that is for a particular factor);
- we sum these variances for all the columns/factors

By doing this **we maximize the sum of the variances inside of each column/factor**, and in essence we'll obtain larger weights on some factor and lower weights on other ones (as opposed to weights all similar). Maximizing it causes the large coefficients to become larger and the small coefficients to approach 0.

Remark 70. If the original solution obtained is difficult to interpret, we can apply this transformation to ease/optimize it.

4.7 Factor score estimation

Important remark 64. Remembering our factor model was

$$\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u}$$

now we are able to estimate $\mathbf{\Lambda}$ by $\hat{\mathbf{\Lambda}}$ (using ML or the other). Now we move to estimation of latent factors values that is $\mathbf{f}$, in order to end with an estimate of $\mathbf{u}$.

Remark 71. After the factor loadings and the unique variances have been estimated, we might be interested in estimating, for each statistical unit whose observed vector is $\mathbf{x}$, the corresponding vector of factor scores $\mathbf{f}$. If for instance the first factor is intelligence, this could also allow us to rank the individuals according to their scores on this factor, from the most to the least intelligent ones. Two methods exist in popular use for factor score estimation.

4.7.1 Thompson estimator

Remark 72. The method proposed by Thompson defines the factor scores as linear combinations of the observed variables chosen so as to minimize the squared expected prediction error.

Remark 73. Scores for factor k-th, named f_k^*, is obtainable as linear combination of value of $\mathbf{x}$, that is $\mathbf{a}_k\mathbf{x}$: if that is so one can choose measure how good is f_k^* in predicting f_k.

According to Thompson's approach we want to find f_k^* such as :

$$\mathbb{E}\left[(f_k^* - f_k)^2\right] \quad \text{is the smallest}$$

This quantity is a function of $\mathbf{a}_k$

$$\phi = \mathbb{E}\left[(\mathbf{a}_k^\top \mathbf{x} - f_k)^2\right]$$

we want to find $\mathbf{a}_k$ for which ϕ is minimum; so go with derivatives and set $= \mathbf{0}$

$$\begin{aligned}\frac{\partial \phi}{\partial \mathbf{a}_k} &= \mathbb{E}\left[2\mathbf{x}(\mathbf{x}^\top \mathbf{a}_k - f_k)\right] = 2\,\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]\mathbf{a}_k - 2\,\mathbb{E}\left[\mathbf{x}f_k\right] \\ &\overset{(1)}{=} 2\mathbf{\Sigma}\mathbf{a}_k - 2\mathbf{\Lambda}_k = 2(\mathbf{\Sigma}\mathbf{a}_k - \mathbf{\Lambda}_k)\end{aligned}$$

where in (1):

- $\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]$ is covariance matrix of $\mathbf{x}$ ($\mathbf{\Sigma}x$)
- with mean centered data $\mathbb{E}\left[\mathbf{x}f_k\right]$ is variance/covariance between $\mathbf{x}$ and f_k thus the k-th column of $\mathbf{\Lambda}$ (call it $\mathbf{\Lambda}_k$).

By putting $2(\mathbf{\Sigma}\mathbf{a}_k - \mathbf{\Lambda}_k) = \mathbf{0}$ and solving for $\mathbf{a}_k$ we obtain

$$\mathbf{a}_k = \mathbf{\Sigma}^{-1}\mathbf{\Lambda}_k$$

Thus:

$$f_k^* = \mathbf{a}_k^\top \mathbf{x} = \mathbf{\Lambda}_k^\top \mathbf{\Sigma}^{-1}\mathbf{x}$$

If we procede like that for all the factor $k = 1, \dots, m$ and collect the results in a vector we obtain

$$\mathbf{f}^* = \mathbf{\Lambda}^\top \mathbf{\Sigma}^{-1} \mathbf{x} \tag{4.5}$$

which is *Thompson estimator* for factor scores. It can be proved that $\mathbf{f}^*$ can equivalently be (sometimes used in books) rewritten as

$$\mathbf{f}^* = \left(\mathbf{I} + \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{\Lambda}\right)^{-1} \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{x} \tag{4.6}$$

Properties of estimator What are the property of this estimator?

- minimize the squared prediction error (by definition/derivation): it has been obtained by minimizing the expected squared prediction error.
- it's biased: if we compute the expected value

$$\begin{aligned}\mathbb{E}\left[\mathbf{f}^* | \mathbf{f}\right] &= \mathbb{E}\left[(\mathbf{I} + \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{\Lambda})^{-1} \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{x} | \mathbf{f}\right] \\ &= \underbrace{(\mathbf{I} + \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{\Lambda})^{-1} \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}}_{\text{all constants}} \cdot \mathbb{E}\left[\mathbf{x} | \mathbf{f}\right]\end{aligned}$$

  but considering $\mathbf{x} = \mathbf{\Lambda}\mathbf{f} + \mathbf{u}$

$$\mathbb{E}\left[\mathbf{x} | \mathbf{f}\right] = \mathbb{E}\left[\mathbf{\Lambda}\mathbf{f} + \mathbf{u} | \mathbf{f}\right] = \mathbf{\Lambda}\mathbf{f} + \underbrace{\mathbb{E}\left[\mathbf{u}\right]}_{=\mathbf{0}} = \mathbf{\Lambda}\mathbf{f}$$

  with $\mathbf{\Lambda}\mathbf{f}$ constant/put out of expectation since conditioning. So coming back we have that

$$\mathbb{E}\left[\mathbf{f}^* | \mathbf{f}\right] = \underbrace{(\mathbf{I} + \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{\Lambda})^{-1} \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \mathbf{\Lambda}}_{\neq \mathbf{I}} \mathbf{f} \neq \mathbf{f}$$

  With the simpler formulation

$$\mathbb{E}\left[\mathbf{f}^* | \mathbf{f}\right] = \mathbb{E}\left[\mathbf{\Lambda}^\top \mathbf{\Sigma}^{-1} \mathbf{x} | \mathbf{f}\right] = \mathbf{\Lambda}^\top \mathbf{\Sigma}^{-1} \mathbb{E}\left[\mathbf{x} | \mathbf{f}\right] = \mathbf{\Lambda}^\top \mathbf{\Sigma}^{-1} \mathbf{\Lambda}\mathbf{f} \neq \mathbf{f}$$

  So $\mathbb{E}\left[\mathbf{f}^* | \mathbf{f}\right] \neq \mathbf{f}$ and this estimator is biased

- Thompson's estimator is also known in the literature as "regression estimator". If we assume that the observed variables $\mathbf{x}$, the common factors $\mathbf{f}$ and the unique factors $\mathbf{u}$ are normally distributed and consider the vector $\mathbf{y}^\top = (\mathbf{f}^\top, \mathbf{x}^\top)$ formed by compounding the common factors and the observed variables, under the factor model assumptions, we obtain that $\mathbf{y}$ is also distributed according to a $\mathbf{0}$ mean multivariate normal distribution with covariance matrix given by

$$\begin{bmatrix} I & \mathbf{\Lambda}^\top \\ \mathbf{\Lambda} & \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi} \end{bmatrix}$$

  From the properties of the multivariate normal distribution, the expected value of $\mathbf{f}$ given $\mathbf{x}$ is

$$\mathbb{E}\left[\mathbf{f} | \mathbf{x}\right] = \mathbf{\Lambda}^\top (\mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi})^{-1} \mathbf{x} = \mathbf{\Lambda}^\top \mathbf{\Sigma}^{-1} \mathbf{x}$$

  that coincides with the previously obtained $\mathbf{f}^*$. Cioè presumo che $\mathbf{\Lambda}^\top \mathbf{\Sigma}^{-1}$ si possa vedere come l'equivalente di una sorta di coefficienti $\boldsymbol{\beta}$ di un modello di regressione che predice $\mathbf{f}$.

4.7.2 Bartlett estimator

Another solution for estimating $\mathbf{f}$ resorting to Bartlett's estimator.

Remark 74. After the factor loadings and the unique variances have been estimated, the factor model can be seen as a linear multivariate regression model where $\mathbf{f}$ is the unknown vector parameter and the residuals (i.e. the unique factors) are uncorrelated but heteroscedastic. Estimation can be addressed by weighted least squares.
In other words, our model is still $\mathbf{x} = \mathbf{\Lambda f} + \mathbf{u}$ and $\text{Var}[\mathbf{u}] = \mathbf{\Psi}$. If estimate $\mathbf{\Lambda}$ using $\hat{\mathbf{\Lambda}}$ this looks like a linear regression model with uncorrelated but heteroscedastic (different variance) error[8]. With non omoschedasticity we can estimate it use *weighted least square* (weight: residual with large variance have lower weight)

We want to find an estimate of $\mathbf{f}$ say $\hat{\mathbf{f}}$, such that the following expression is minimized

$$\mathbf{u}^\top \mathbf{\Psi}^{-1}\mathbf{u} = (\mathbf{x} - \mathbf{\Lambda f})^\top \mathbf{\Psi}^{-1}(\mathbf{x} - \mathbf{\Lambda f})$$

Deriving with respect of $\mathbf{f}$ we obtain

$$\frac{\partial}{\partial \mathbf{f}} = -2\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}(\mathbf{x} - \mathbf{\Lambda f}) = -2\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}x + 2\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda f}$$

Setting the last $= \mathbf{0}$ we have:

$$\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda f} = \mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{x}$$

in order to obtain $\mathbf{f}$ we need to calculate the inverse of $\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda}$. So the Bartlett estimator is

$$\hat{\mathbf{f}} = (\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{x} \tag{4.7}$$

Properties This estimator

- unbiased since

$$\begin{aligned}
\mathbb{E}\left[\hat{\boldsymbol{f}}|\mathbf{f}\right] &= \mathbb{E}\left[(\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{x}|\mathbf{f}\right] \\
&= (\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1} \cdot \underbrace{\mathbb{E}\left[\mathbf{x}|\mathbf{f}\right]}_{\mathbf{\Lambda f}} \\
&= (\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}^\top \mathbf{\Psi}^{-1}\mathbf{\Lambda f} \\
&= \mathbf{I f} = \mathbf{f}
\end{aligned}$$

- compared to the previous one, however, has larger variance: previous has the lowest variance but has bias.

The choice between the two is dependent on the research goals.

[8] in regression residuals were expected to be omoschedastic ($\text{Var}[\boldsymbol{\varepsilon}] = \sigma^2\mathbf{I}$)

4.8 Relationship between PCA and Factor Analysis

Remark 75. It is worth concluding this chapter by stressing the connections and the differences between Factor Analysis and PCA. Both methods have the aim of reducing the dimensionality of a vector of random variables. But while Factor Analysis assumes a model (that may fit the data or not), PCA is just a data transformation and for this reason it always exists. Furthermore while Factor Analysis aims at explaining (covariances) or correlations, PCA concentrates on variances.

Despite these conceptual differences, there have been attempts, mainly in the past, to use PCA in order to estimate the factor model. In the following we will show that indeed PCA may be inadequate when the goal of the research is fitting a factor model.

Important remark 65. The principal factor method perform spectral decomposition on the reduced covariance matrix, defined as

$$\mathbf{S} - \hat{\mathbf{\Psi}} = \begin{bmatrix} h_1^2 & s_{12} & \dots & s_{1p} \\ & h_2^2 & \dots & s_{2p} \\ & & \dots & \\ & & & h_p^2 \end{bmatrix}$$

If, as seen in section 4.5, as first communality we set:

- $h_i^2 = \sigma_i^2$ (for unstandardized data) the reduced covariance matrix is just the observed one $\mathbf{S}$, according to our choice for the communalities $\mathbf{S} - \hat{\mathbf{\Psi}} = \mathbf{S}$. Extracting factor using principal factor method on $\mathbf{S} - \hat{\mathbf{\Psi}}$ amounts to perform the spectral decomposition of $\mathbf{S}$, that is to obtain PCs.

- $h_i^2 = 1$ (for unstandardized data, using $\mathbf{R}$ instead of $\mathbf{S}$ above we have correlation coefficients out of main diagonal) the reduced covariance (correlation) matrix coincides with the original correlation matrix. Thus again we are performing PCA on the $\mathbf{R} - \hat{\mathbf{\Psi}} = \mathbf{R}$ original correlation matrix

On the contrary assume we have $\mathbf{x}$ which is $p \times 1$ and obtained principal components $\mathbf{y} = \mathbf{A}^\top \mathbf{x}$ with $\mathbf{A}$ the orthonormal matrix whose columns are the eigenvectors of the covariance matrix of the $\mathbf{x}$ variables.
Since $\mathbf{A}$ is orthogonal $\mathbf{A}^{-1} = \mathbf{A}^\top$ and by pre-multiplying by $\mathbf{A}^{-1}$ we obtain:

$$\mathbf{x} = \mathbf{A}\mathbf{y}$$

I can split matrix $\mathbf{A}$ into two parts

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_m & \mathbf{A}_{p-m} \end{bmatrix}$$

with $\mathbf{A}_m$ the first m eigenvector (corresponding to the first m eigenvalues) and $\mathbf{A}_{p-m}$ the remaining ones. $\mathbf{y}$ can be splitted vertically

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_m \\ \mathbf{y}_{p-m} \end{bmatrix}$$

this means that $\mathbf{x}$ can be written as

$$\mathbf{x} = \mathbf{A}\mathbf{y} = \begin{bmatrix}\mathbf{A}_m & \mathbf{A}_{p-m}\end{bmatrix} \begin{bmatrix}\mathbf{y}_m \\ \mathbf{y}_{p-m}\end{bmatrix} = \mathbf{A}_m\mathbf{y}_m + \mathbf{A}_{p-m}\mathbf{y}_{p-m}$$

We can also put $\mathbf{I}$ in the right place and decompose it as $\mathbf{L}_m^{1/2}\mathbf{L}_m^{-1/2}$ (with $\mathbf{L}_m$ a diagonal matrix with the first m eigenvalues)

$$\begin{aligned}\mathbf{x} &= \mathbf{A}_m\mathbf{I}_m\mathbf{y}_m + \mathbf{A}_{p-m}\mathbf{y}_{p-m} \\ &= \mathbf{A}_m\mathbf{L}_m^{1/2}\mathbf{L}_m^{-1/2}\mathbf{y}_m + \mathbf{A}_{p-m}\mathbf{y}_{p-m}\end{aligned}$$

In so doing I separated first m principal components on one side and the remaining on the other size. Now let's call

$$\begin{aligned}\mathbf{A}_m\mathbf{L}_m^{1/2} &= \boldsymbol{\Lambda} \\ \mathbf{L}_m^{-1/2}\mathbf{y}_m &= \mathbf{f} \\ \mathbf{A}_{p-m}\mathbf{y}_{p-m} &= \boldsymbol{\eta}\end{aligned}$$

So we have that the above can be rewritten as

$$\mathbf{x} = \boldsymbol{\Lambda}\mathbf{f} + \boldsymbol{\eta}$$

Is this a linear factor model? if the answer is yes PCA is a special case of factor analysis, otherwise they are two different things. The answer is that **this is not a linear factor model**.
To check it we check whether latent variables involved in the model satisfy all the assumption; especially we want to check the linear assumptions we made

1. is true that $\mathbb{E}[\mathbf{f}] = \mathbf{0}$? we have that

   $$\mathbb{E}\left[\mathbf{L}_m^{-1/2}\mathbf{Y}_m\right] = \mathbf{0}$$

   because we're working with mean centered data. So this first condition is satisfied

2. is common factor uncorrelated and have unit variance, that is $\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right] = \mathbf{I}$? we have

   $$\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right] = \mathbb{E}\left[\mathbf{L}_m^{-1/2}\mathbf{y}_m\mathbf{y}_m^\top\mathbf{L}_m^{-1/2}\right] \overset{(1)}{=} \mathbf{L}_m^{-1/2}\,\mathbb{E}\left[\mathbf{y}_m\mathbf{y}_m^\top\right]\mathbf{L}_m^{-1/2}$$

   where in (1) we take $\mathbf{L}_m^{-1/2}$ out of expectation. What is $\mathbb{E}\left[\mathbf{y}_m\mathbf{y}_m^\top\right]$? It's the variance/covariance of the first m principal components, a diagonal matrix with eigenvalues on diagonal, $\mathbf{L}_m$. So we have

   $$\mathbb{E}\left[\mathbf{f}\mathbf{f}^\top\right] = \mathbf{L}_m^{-1/2}\mathbf{L}_m\mathbf{L}_m^{-1/2} = \mathbf{L}_m^{-1/2}\mathbf{L}_m^{1/2}\mathbf{L}_m^{1/2}\mathbf{L}_m^{-1/2} = \mathbf{I}\mathbf{I} = \mathbf{I}$$

   so OK, the property is verified

3. is $\mathbb{E}[\boldsymbol{\eta}] = \mathbf{0}$? We have

   $$\mathbb{E}[\boldsymbol{\eta}] = \mathbb{E}[\mathbf{A}_{p-m}\mathbf{y}_{p-m}] = \mathbf{A}_{p-m}\,\mathbb{E}[\mathbf{y}_{p-m}] \overset{(1)}{=} \mathbf{0}$$

   where in (1) we have that $\mathbb{E}[\mathbf{Y}_{p-m}] = \mathbf{0}$ because mean centered.
   So even this this is satisfied

4. property fundamental which relates common factor and unique factors, that is $\mathbb{E}\left[\mathbf{f}^\top \boldsymbol{\eta}\right] = \mathbf{0}$

$$\mathbb{E}\left[\mathbf{f}^\top \boldsymbol{\eta}\right] = \mathbb{E}\left[\mathbf{f}\boldsymbol{\eta}^\top\right] = \mathbb{E}\left[\mathbf{L}_m^{-1/2}\mathbf{y}_m\mathbf{y}_{p-m}^\top\mathbf{A}_{p-m}^\top\right] = \mathbf{L}_m^{-1/2}\,\mathbb{E}\left[\mathbf{y}_m\mathbf{y}_{p-m}^\top\right]\mathbf{A}_{p-m}^\top$$
$$= \mathbf{0}$$

where $\mathbb{E}\left[\mathbf{y}_m\mathbf{y}_{p-m}^\top\right]$ is the covariance between the first m components and the last $p-m$ ones. This is $\mathbf{0}$ because PCs are uncorrelated so actually all is 0 and check is verified;

5. is $\mathbb{E}\left[\boldsymbol{\eta}\boldsymbol{\eta}^\top\right]$ a diagonal matrix similar to

$$\boldsymbol{\Psi} = \begin{bmatrix} \psi_{11} & \dots & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & \psi_{ii} & \dots & 0 \\ \dots & & & & \\ 0 & \dots & 0 & \dots & \psi_{pp} \end{bmatrix}$$

We have

$$\mathbb{E}\left[\boldsymbol{\eta}\boldsymbol{\eta}^\top\right] = \mathbb{E}\left[\mathbf{A}_{p-m}\mathbf{y}_{p-m}\mathbf{y}_{p-m}^\top\mathbf{A}_{p-m}^\top\right] = \mathbf{A}_{p-m}\,\mathbb{E}\left[\mathbf{y}_{p-m}\mathbf{y}_{p-m}^\top\right]\mathbf{A}_{p-m}^\top$$
$$\overset{(1)}{=} \mathbf{A}_{p-m}\mathbf{L}_{p-m}\mathbf{A}_{p-m}^\top$$

where in (1) $\mathbb{E}\left[\mathbf{y}_{p-m}\mathbf{y}_{p-m}^\top\right] = \mathbf{L}_{p-m}$ is the variance covariance of last $p-m$ PCs; it's diagonal with last $p-m$ eigenvalues. However if these entries/diagonal elements are different this is not enough so the whole product $\mathbb{E}\left[\boldsymbol{\eta}\boldsymbol{\eta}^\top\right]$ is not diagonal.
Thus the condition does not hold: the unique factor $\boldsymbol{\eta}$ are not uncorrelated, and this means that in the model

$$\mathbf{x} = \boldsymbol{\Lambda}\mathbf{f} + \boldsymbol{\eta}$$

the common factors $\mathbf{f}$ are not able to explain the observed covariances (which was the goal of fitting a factor model).

Important remark 66 (Take home message). If we decide to put communality = 1 or observed variances the algorithm fits principal components, but this is not a model: its just PCA because it does not satisfy all the requirements of Factor Analysis.
People wrongly think that what can be done in FA can be done in PCA (eg rotating); this is just wrong. The property of Linear factor model does not holds with PCA.

4.9 Lab

Factor Analysis (FA) is a multivariate statistical technique that aims to explain the covariances (or correlations) of a large number of variables by using a lower number of latent factors. In this sense, FA can be viewed as a tool that allows to reduce the dimensionality of a given dataset.

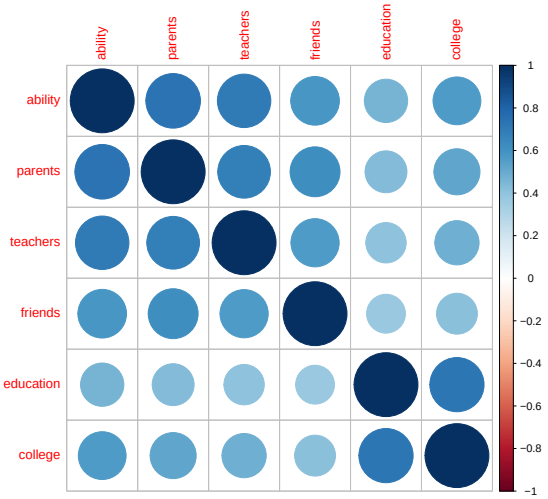
4.9.1 Example 1 (correlation matrix)

The following data were taken from the book Applied Multivariate Data Analysis (Everitt & Dunn 2001), Section 12.5. They refer to the following six variables:

- self-concept of ability
- perceived parental evaluation
- perceived teacher evaluation
- perceived friend's evaluation
- educational aspiration
- college plans

The units are 556 primary school children; we have just a correlation matrix, that we transform into a matrix

```
if (FALSE) ability <- read.table("multivariate_statistics/data/ability-education.txt")
ability <- read.table("data/ability-education.txt")

n <- 556
(R <- as.matrix(ability))

##           ability parents teachers friends education college
## ability      1.00    0.73     0.70    0.58      0.46    0.56
## parents      0.73    1.00     0.68    0.61      0.43    0.52
## teachers     0.70    0.68     1.00    0.57      0.40    0.48
## friends      0.58    0.61     0.57    1.00      0.37    0.41
## education    0.46    0.43     0.40    0.37      1.00    0.72
## college      0.56    0.52     0.48    0.41      0.72    1.00

corrplot::corrplot(R) ## useful for higher number of variables
```



We have quite high correlation all positive, it makes sense to perform factor analysis: in order to reduce the dimension of the problem and to explain the observed correlations through some related latent factors we fit a factor model using the principal factor method.

4.9.1.1 Principal factor analysis by matrix functions

Performing calculations by hand require starting from a preliminary value of communality h_i^2 for each variable $i = 1, \ldots, p$. A reasonal approximation for this value is provided by the multiple correlation coefficient:

$$R_{i0}^2 = 1 - \frac{1}{r_{ii}^*}$$

where r_{ii}^* is the i-th diagonal element of $\mathbf{R}^{-1}$

```
(h2tilde <- 1 - 1 / diag(solve(R)))

##   ability   parents  teachers   friends education   college
## 0.6427569 0.6248924 0.5695938 0.4358076 0.5265227 0.5928205
```

The second step involves the computation of the reduced correlation matrix $\mathbf{R}^* = \mathbf{R} - \mathbf{\Psi}$ by substituting the estimated communalities to the diagonal elements (the 1's) of the original correlation matrix:

```
redR <- R # replace diag with preliminary estimates of communalities
diag(redR) <- h2tilde
redR # no more 1 on diagonal

##             ability   parents  teachers   friends education   college
## ability   0.6427569 0.7300000 0.7000000 0.5800000 0.4600000 0.5600000
## parents   0.7300000 0.6248924 0.6800000 0.6100000 0.4300000 0.5200000
## teachers  0.7000000 0.6800000 0.5695938 0.5700000 0.4000000 0.4800000
## friends   0.5800000 0.6100000 0.5700000 0.4358076 0.3700000 0.4100000
## education 0.4600000 0.4300000 0.4000000 0.3700000 0.5265227 0.7200000
## college   0.5600000 0.5200000 0.4800000 0.4100000 0.7200000 0.5928205
```

To compute factor loadings we need to compute a low rank (where the rank is the number of factors we want) approximation of the reduced correlation matrix $\mathbf{R} = \mathbf{\Gamma L \Gamma}^\top$

```
## to do SVD
spectral <- eigen(redR)
spectral$values # eigenvalues

## [1]  3.33126929  0.47350875 -0.04368765 -0.08468263 -0.11304912 -0.17096486
```

we see only first two eigenvalues are positive: this because reduced correlation matrix is not positive definite. Looking at eigenvalues we can think considering that two factor should be enough to explain the correlations of the original data, m should be 2.
Thus we build the matrix of eigenvector with just 2 eigenvectors and vectors of eigenvalue with just 2 eigenvalues

```
G <- spectral$vectors[, 1:2]     # gamma only hte first two columns/eigvenvector
L <- diag(spectral$values[1:2]) # diag matrix with first two eigenvalues
G %*% L %*% t(G)  # low rank approximation of correlation matrix

##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.7072292 0.7001694 0.6668653 0.5834546 0.4685601 0.5441205
## [2,] 0.7001694 0.6955646 0.6634608 0.5801711 0.4374113 0.5137057
## [3,] 0.6668653 0.6634608 0.6332413 0.5536204 0.4057116 0.4789896
## [4,] 0.5834546 0.5801711 0.5536204 0.4840494 0.3583494 0.4222716
## [5,] 0.4685601 0.4374113 0.4057116 0.3583494 0.6043086 0.6378458
## [6,] 0.5441205 0.5137057 0.4789896 0.4222716 0.6378458 0.6803849
```

Now we can compute the estimate of the factor loading matrix $\hat{\mathbf{\Lambda}} = \mathbf{\Gamma}_m \mathbf{L}_m^{1/2}$ where $\mathbf{L}_m$ is the diagonal matrix containing only the m positive eigenvalues of `redR` and $\mathbf{\Gamma}_m$ is the matrix containing by columns the corresponding eigenvectors.

```
(Lambda <- G %*% L^0.5)  # factor loadings

##            [,1]       [,2]
## [1,] -0.8273744 -0.1506015
## [2,] -0.8103705 -0.1971401
## [3,] -0.7682259 -0.2075338
## [4,] -0.6732548 -0.1754348
## [5,] -0.6452398  0.4335599
## [6,] -0.7281779  0.3874814
```

It's a matrix of dimension 6x2 is what we want: 2 column (1 per factor), 6 variables in row(we'll interpret later).
Now we compute the new communality: communality is part of variance of a variable explained by the common factors. To extract communality, the sum of squares of the i-th row of $\hat{\mathbf{\Lambda}}$ is the communality of the i-th observed variable. Two methods actually:

```
## Lambda t(Lambda) and extract diagonal (interested only in the variances not
## covariances)
(communalities <- diag(Lambda %*% t(Lambda)))

## [1] 0.7072292 0.6955646 0.6332413 0.4840494 0.6043086 0.6803849

## or equivalently
apply(Lambda^2, 1, sum)

## [1] 0.7072292 0.6955646 0.6332413 0.4840494 0.6043086 0.6803849
```

These show, for example, that the 71% of variance in self-concept of ability is explained by the two common factors. Of course, the larger the communality the better the variable does serve as an indicator of the associated factors.
Finally, the goodness of fit of the model can be evaluated through the residual correlation matrix $\mathbf{R} - \mathbf{\Lambda}\mathbf{\Lambda}^\top$; this is theoretically a diagonal matrix, so by

looking at its off-diagonal entries (how far they are from 0), can give an idea of the model fit.

```
R - Lambda %*% t(Lambda) # out of main diagonal should be close to zero

##                ability      parents     teachers      friends    education
## ability    0.292770802  0.029830608  0.033134677 -0.003454555 -0.008560141
## parents    0.029830608  0.304435440  0.016539177  0.029828913 -0.007411273
## teachers   0.033134677  0.016539177  0.366758716  0.016379562 -0.005711590
## friends   -0.003454555  0.029828913  0.016379562  0.515950562  0.011650634
## education -0.008560141 -0.007411273 -0.005711590  0.011650634  0.395691356
## college    0.015879548  0.006294259  0.001010408 -0.012271579  0.082154195
##                college
## ability    0.015879548
## parents    0.006294259
## teachers   0.001010408
## friends   -0.012271579
## education  0.082154195
## college    0.319615088

## test mio
# corrplot::corrplot(R - Lambda %*% t(Lambda)) ## out of diagonal low values
```

If elements out of the main diagonal are close to 0, then factors are able to well capture the correlations between the original variables. Since in this case the elements out of the diagonal are fairly small and close to zero we can conclude that the model fits adequately the data.
Some remaining stuff:

- the uniqueness: diagonal of correlation matrix - communality

```
(uniqueness <- diag(R) - communalities) # == 1 - communalities

##   ability   parents  teachers   friends education   college
## 0.2927708 0.3044354 0.3667587 0.5159506 0.3956914 0.3196151
```

- variance explained by two factors; remember that here total variability is p. To do that we use the diagonal (interested just in the variance, not covariance) of $\mathbf{\Lambda}^\top \mathbf{\Lambda}$ and ratio it to

```
diag(t(Lambda) %*% Lambda) # interested in diagonal, the variance

## [1] 3.3312693 0.4735087

## so the first factor explain 3.33 and second 0.47

## percentage explained
diag(t(Lambda) %*% Lambda) / 6 # number of variables

## [1] 0.55521155 0.07891812

## first factor explain 55% of total variability, 7% for the second
```

4.9.1.2 Performing FA with `factanal` and `fa`

`psych::fa` gives both principal factor method and ML based estimation: as main arguments it requires

- `r`: A correlation or covariance matrix or a raw data matrix. If covariances are provided, they will be converted to correlations if `covar=FALSE`, while they will be kept as covariances if `covar=TRUE`;
- `nfactors`: specify the number of factors you want;
- `fm`: specify the factoring method you want to perform. In particular, `fm="pa"` will do the principal factor solution, while `fm="ml"` will do a maximum likelihood factor analysis. The default method `minres` has not be seen in class;
- `rotate`: you need to specify "none" or the name of the function to be used to rotate the factors;
- `scores`: "regression" finds factor scores using regression. Among the alternatives there is "Bartlett";
- `n.obs`: the number of observations used to find the correlation matrix if you are providing as input a correlation matrix. It should be needed for p-values, we're not using it

```
library(psych)

##
## Caricamento pacchetto:  'psych'
## Il seguente oggetto è mascherato da 'package:car':
##
##     logit
## Il seguente oggetto è mascherato da 'package:lbmisc':
##
##     table2df
```

```
(pfa1 <- fa(R,
            nfactors = 2, # nfactors wanted (we already seen two positive eigenv)
            rotate = "none", # for the moment we dont' rotatae
            fm = "pa")) # principal factor solution as done before
```

```
## Factor Analysis using method =  pa
## Call: fa(r = R, nfactors = 2, rotate = "none", fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##            PA1   PA2   h2   u2 com
## ability   0.83 -0.18 0.73 0.27 1.1
## parents   0.82 -0.24 0.73 0.27 1.2
## teachers  0.77 -0.24 0.66 0.34 1.2
## friends   0.67 -0.19 0.49 0.51 1.2
## education 0.67  0.48 0.68 0.32 1.8
## college   0.76  0.45 0.77 0.23 1.6
```

```
##
##                        PA1  PA2
## SS loadings           3.44 0.61
## Proportion Var        0.57 0.10
## Cumulative Var        0.57 0.68
## Proportion Explained  0.85 0.15
## Cumulative Proportion 0.85 1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 2 factors are sufficient.
##
## df null model =  15  with the objective function =  3.3
## df of  the model are 4  and the objective function was  0.01
##
## The root mean square of the residuals (RMSR) is  0.01
## The df corrected root mean square of the residuals is  0.02
##
## Fit based upon off diagonal values = 1
## Measures of factor score adequacy
##                                                 PA1  PA2
## Correlation of (regression) scores with factors   0.96 0.83
## Multiple R square of scores with factors          0.92 0.68
## Minimum correlation of possible factor scores     0.84 0.36
```

We see that SS loadings are total variability explained second row is the proportion of variability.
Loadings estimates are similar (up to sign) to what got before by hand

```
pfa1$loadings # matrix of loadings
```

```
##
## Loadings:
##           PA1    PA2
## ability    0.834 -0.181
## parents    0.822 -0.236
## teachers   0.775 -0.239
## friends    0.672 -0.192
## education  0.666  0.483
## college    0.757  0.445
##
##                  PA1   PA2
## SS loadings    3.440 0.614
## Proportion Var 0.573 0.102
## Cumulative Var 0.573 0.676
```

```
Lambda # similar a part of the sign to what obtained before
```

```
##            [,1]       [,2]
## [1,] -0.8273744 -0.1506015
## [2,] -0.8103705 -0.1971401
```

```
## [3,] -0.7682259 -0.2075338
## [4,] -0.6732548 -0.1754348
## [5,] -0.6452398  0.4335599
## [6,] -0.7281779  0.3874814
```

Communalities are similar to what seen before

```
pfa1$communality # similar

##   ability   parents  teachers   friends education   college
## 0.7292947 0.7305715 0.6570953 0.4882353 0.6776897 0.7714643

## now communalities derived from the loadings
hatLambda <- pfa1$loadings
diag(hatLambda %*% t(hatLambda))

##   ability   parents  teachers   friends education   college
## 0.7292947 0.7305715 0.6570953 0.4882353 0.6776897 0.7714643
```

Uniqueness

```
pfa1$uniquenesses

##   ability   parents  teachers   friends education   college
## 0.2707053 0.2694285 0.3429047 0.5117647 0.3223103 0.2285357
```

Residual correlation matrix

```
pfa1$residual # off diagonal still values very close to 0 (good fit)

##                ability      parents      teachers      friends     education
## ability    0.270705253  0.001639509  0.0102526111 -0.015507856 -0.0084315471
## parents    0.001639509  0.269428492 -0.0127285853  0.012765292 -0.0034115950
## teachers   0.010252611 -0.012728585  0.3429047263  0.003724221 -0.0007200665
## friends   -0.015507856  0.012765292  0.0037242208  0.511764685  0.0147570912
## education -0.008431547 -0.003411595 -0.0007200665  0.014757091  0.3223103254
## college    0.008871656  0.002985356 -0.0001766904 -0.013570761  0.0002745308
##                 college
## ability    0.0088716563
## parents    0.0029853559
## teachers  -0.0001766904
## friends   -0.0135707613
## education  0.0002745308
## college    0.2285356519
```

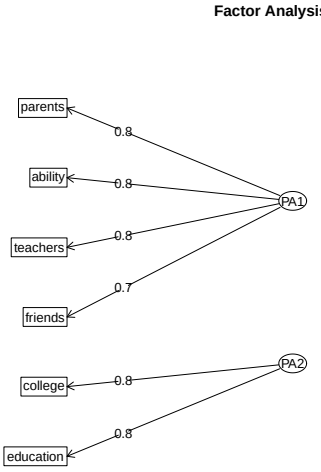
Since loadings are not rotatad this can be difficult to be interpret, if we rotate it helps

```
(pfa2 <- fa(R,
            nfactors = 2,
            rotate = "varimax", # unique change
            fm = "pa"))
```

```
## Factor Analysis using method =  pa
## Call: fa(r = R, nfactors = 2, rotate = "varimax", fm = "pa")
## Standardized loadings (pattern matrix) based upon correlation matrix
##            PA1  PA2   h2   u2 com
## ability   0.79 0.33 0.73 0.27 1.3
## parents   0.81 0.28 0.73 0.27 1.2
## teachers  0.77 0.25 0.66 0.34 1.2
## friends   0.66 0.23 0.49 0.51 1.2
## education 0.27 0.78 0.68 0.32 1.2
## college   0.36 0.80 0.77 0.23 1.4
##
##                        PA1  PA2
## SS loadings           2.50 1.56
## Proportion Var        0.42 0.26
## Cumulative Var        0.42 0.68
## Proportion Explained  0.62 0.38
## Cumulative Proportion 0.62 1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 2 factors are sufficient.
##
## df null model =  15  with the objective function =  3.3
## df of  the model are 4  and the objective function was  0.01
##
## The root mean square of the residuals (RMSR) is  0.01
## The df corrected root mean square of the residuals is  0.02
##
## Fit based upon off diagonal values = 1
## Measures of factor score adequacy
##                                                  PA1  PA2
## Correlation of (regression) scores with factors   0.91 0.88
## Multiple R square of scores with factors          0.83 0.77
## Minimum correlation of possible factor scores     0.66 0.54
```

Looking at the loadings, the first factor has high entries for first four variables (self-perceived evaluation of their environment and themselves - in the book where the example is taken it is termed as "ability"- let's call it perceived ability), while the second has high loading for the last variable (could be seen as a latent variable relating to children's attitude towards education). To visualize the factor

```
# visualization of factor
fa.diagram(pfa2)
```



ML estimation

Distributional assumptions on $\mathbf{x}$:

$$\mathbf{x} \sim \text{MVN}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}\right)$$

There are two functions implemented in R that can be used to perform the maximum likelihood estimation of the factor model:

- `psych::fa` function with `fm = "ml"`
- `stats::factanal`,

For the estimation with `fa`

```
(mlfa1 <- fa(R,
             nfactors = 2,
             rotate = "none",
             fm = "ml" )) # only change

## Factor Analysis using method =  ml
## Call: fa(r = R, nfactors = 2, rotate = "none", fm = "ml")
## Standardized loadings (pattern matrix) based upon correlation matrix
##             ML1   ML2   h2    u2 com
## ability    0.57  0.64 0.73 0.266 2.0
## parents    0.54  0.66 0.73 0.274 1.9
## teachers   0.49  0.64 0.66 0.344 1.9
## friends    0.42  0.56 0.49 0.509 1.9
## education 0.72  0.07 0.53 0.471 1.0
## college    1.00 -0.02 1.00 0.005 1.0
##
##                        ML1  ML2
## SS loadings           2.56 1.57
```

```
## Proportion Var        0.43 0.26
## Cumulative Var        0.43 0.69
## Proportion Explained  0.62 0.38
## Cumulative Proportion 0.62 1.00
##
## Mean item complexity =  1.6
## Test of the hypothesis that 2 factors are sufficient.
##
## df null model =  15  with the objective function =  3.3
## df of  the model are 4  and the objective function was  0.01
##
## The root mean square of the residuals (RMSR) is  0.01
## The df corrected root mean square of the residuals is  0.02
##
## Fit based upon off diagonal values = 1
## Measures of factor score adequacy
##                                                  ML1  ML2
## Correlation of (regression) scores with factors   1.00 0.91
## Multiple R square of scores with factors          1.00 0.83
## Minimum correlation of possible factor scores     0.99 0.67

## as before we have
mlfa1$loadings # without rotation

##
## Loadings:
##           ML1    ML2
## ability    0.575  0.635
## parents    0.535  0.663
## teachers   0.495  0.641
## friends    0.423  0.558
## education  0.723
## college    0.997
##
##                  ML1   ML2
## SS loadings    2.558 1.571
## Proportion Var 0.426 0.262
## Cumulative Var 0.426 0.688

mlfa1$communality

##   ability   parents  teachers   friends education   college
## 0.7336711 0.7255385 0.6555405 0.4907396 0.5286846 0.9950001

mlfa1$uniquenesses

##     ability     parents    teachers     friends   education     college
## 0.266328898 0.274461521 0.344459539 0.509260352 0.471315389 0.004999905
```

To use `factanal` arguments are a litte different:

- `x`: a formula or a numeric matrix (it's not correlation matrix, it's the data!) or an object that can be coerced to a numeric matrix ;
- `factors`: the number of factors to be fitted;
- `data`: an optional data frame used only if x is a formula;
- `covmat`: a covariance (or correlation) matrix (if we don't have the data);
- `n.obs`: the number of observations. This argument is needed only if `covmat` is a covariance matrix if we want scores or chisquare p-values;
- `start`: NULL or a matrix of starting values, each column giving an initial set of uniquenesses;
- `scores`: type of scores to produce, if any. The default is none; "regression" gives Thompson's ores, i.e. the linear combinations of the observed variables chosen so as to minimize the squared ediction error; "Bartlett" gives Bartlett's weighted least-squares scores.
- `rotation`: none or the name of the function to be used to rotate the factors, by default `varimax`. If we dont' want to rotate specify `"none"`

```
(mlfactanal1 <- factanal(covmat = R, factors = 2,  n.obs = 556, rotation = "none"))

##
## Call:
## factanal(factors = 2, covmat = R, n.obs = 556, rotation = "none")
##
## Uniquenesses:
##   ability   parents  teachers   friends education   college
##     0.266     0.274     0.344     0.509     0.471     0.005
##
## Loadings:
##           Factor1 Factor2
## ability    0.575   0.635
## parents    0.535   0.663
## teachers   0.495   0.641
## friends    0.423   0.558
## education  0.723
## college    0.997
##
##                Factor1 Factor2
## SS loadings      2.558   1.571
## Proportion Var   0.426   0.262
## Cumulative Var   0.426   0.688
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 4.57 on 4 degrees of freedom.
## The p-value is 0.335
```

Since we've specified `n.obs` we have chisq and p-value: p-value is larger than usual alpha: we can conclude that two factors are enough to explain correlation.

```
## again
mlfactanal1$loadings  # results exactly equal to fa with ml option

##
## Loadings:
##           Factor1 Factor2
## ability    0.575   0.635
## parents    0.535   0.663
## teachers   0.495   0.641
## friends    0.423   0.558
## education  0.723
## college    0.997
##
##                Factor1 Factor2
## SS loadings      2.558   1.571
## Proportion Var   0.426   0.262
## Cumulative Var   0.426   0.688

mlfactanal1$uniquenesses

##   ability   parents  teachers   friends education   college
## 0.2663356 0.2744603 0.3444599 0.5092609 0.4713136 0.0050000

1 - mlfactanal1$uniquenesses ## communality

##   ability   parents  teachers   friends education   college
## 0.7336644 0.7255397 0.6555401 0.4907391 0.5286864 0.9950000
```

4.9.2 Example 2 (ML FA on dataset with m choosing): athletics data

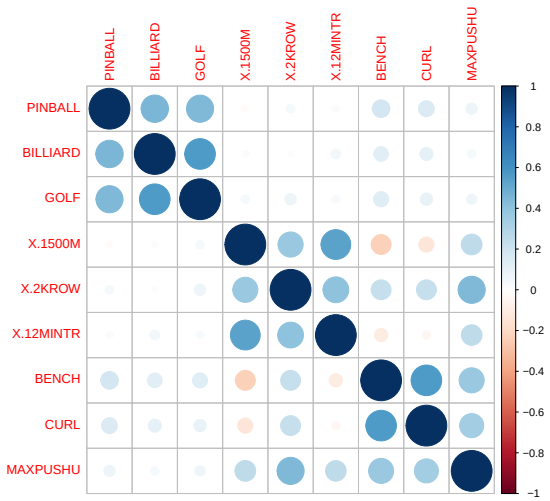
File `AthleticsData.sav` contains measurements over 9 different athletic disciplines recorded on 1000 students. In particular, variables are:

- `PINBALL`
- `BILLIARD`
- `GOLF`
- `X.1500M`, i.e. 1500 mt run
- `X.2KROW`, i.e. 2 km row
- `X.12MINTR`, i.e. 12 minutes run
- `BENCH`
- `CURL`

- MAXPUSHU

The aim of the analysis is to reduce the dimension of the problem by measuring some latent factors that impact students' performances.

```
# cleaning
rm(list = ls())
## import
x <- Hmisc::spss.get("data/AthleticsData.sav")
if (FALSE) x <- Hmisc::spss.get("multivariate_statistics/data/AthleticsData.sav")

head(x)

##      PINBALL     BILLIARD         GOLF    X.1500M    X.2KROW   X.12MINTR
## 1 -1.1225055  0.009316132 -1.52679352 -0.9483176 -0.1647701 -0.05203922
## 2  0.3286001 -0.745125995 -0.84888702  0.6849068  0.1455623  0.13481553
## 3  0.5442109  0.823572688  0.55194360 -0.6842024 -0.5152493 -0.24014598
## 4  1.7282347 -0.142108710  0.95376088  0.9312700 -1.0275236 -0.89791136
## 5  0.8650813  0.363277424 -0.56698861  0.9757308  1.2200180  0.24952087
## 6  0.8441094  0.438722000  0.05975168 -0.6828077  0.6336518  0.30713267
##        BENCH        CURL    MAXPUSHU
## 1  1.3593056 -0.83766332 -0.04783271
## 2 -0.4906018  0.22148232  0.38120977
## 3 -0.8188845 -0.62012251 -1.13213981
## 4 -0.9732271 -1.12976703 -0.33035037
## 5  1.2293106 -0.03462911  0.40703588
## 6  0.1073211 -0.81878026 -0.15481228

dim(x)

## [1] 1000    9

corrplot::corrplot(cor(x))
```



We

- see two blocks of variables with quite high correlation so it make sense to perform factor analysis.
- do not know a priori the correct number of latent factors. Therefore, it is highly suggested to perform FA with different values for m and, then, to choose the most suitable one.

We perform Factor analysis according to a maximum likelihood approach by `factanal`

4.9.2.1 FA with 2 factors

```
(ml2 <- factanal(x, factors = 2, rotation = "none"))

##
## Call:
## factanal(x = x, factors = 2, rotation = "none")
##
## Uniquenesses:
##   PINBALL  BILLIARD      GOLF   X.1500M   X.2KROW X.12MINTR     BENCH      CURL
##     0.938     0.962     0.955     0.361     0.534     0.536     0.301     0.540
##  MAXPUSHU
##     0.560
##
## Loadings:
##           Factor1 Factor2
## PINBALL    0.249
## BILLIARD   0.192
## GOLF       0.206
## X.1500M            0.793
## X.2KROW    0.413   0.544
## X.12MINTR          0.681
## BENCH      0.813  -0.193
## CURL       0.673
## MAXPUSHU   0.545   0.379
##
##                Factor1 Factor2
## SS loadings      1.734   1.579
## Proportion Var   0.193   0.175
## Cumulative Var   0.193   0.368
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 652.4 on 19 degrees of freedom.
## The p-value is 4.3e-126
```

At the end of the output, we can find χ^2-goodness-of-fit statistic (very high) and correponding p-value very small : we reject H_0 that our variables can be explained by two factors/ a 2 factor model well fits the data is rejected.

4.9.2.2 FA with 3 factors

Increase `factors` by 1 and see what happens:

```
(ml3 <- factanal(x, factors = 3, rotation = "none"))

##
## Call:
## factanal(x = x, factors = 3, rotation = "none")
##
## Uniquenesses:
##   PINBALL  BILLIARD      GOLF   X.1500M   X.2KROW X.12MINTR     BENCH      CURL
##     0.635     0.414     0.455     0.361     0.520     0.538     0.302     0.536
##  MAXPUSHU
##     0.540
##
## Loadings:
##           Factor1 Factor2 Factor3
## PINBALL    0.425           0.429
## BILLIARD   0.443           0.624
## GOLF       0.447           0.585
## X.1500M            0.799
## X.2KROW    0.408   0.496  -0.260
## X.12MINTR          0.672
## BENCH      0.729  -0.280  -0.297
## CURL       0.605  -0.158  -0.270
## MAXPUSHU   0.512   0.317  -0.312
##
##                Factor1 Factor2 Factor3
## SS loadings      1.912   1.545   1.243
## Proportion Var   0.212   0.172   0.138
## Cumulative Var   0.212   0.384   0.522
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 12.94 on 12 degrees of freedom.
## The p-value is 0.373
```

Here p-value is larger and we don't reject H_0: 3 factors seem enough for this dataset/to account for the correlations among the original variables.
Looking at the `corrplot` (just a visual tool) it seemed that two were enough, but loking formally there are 3 blocks.
The output reports the uniquenesses, i.e. the variances of the unique factors. Since the algorithm fits the model using the correlation matrix, the communalities can be obtained as 1 minus the corresponding uniquenesses.

```
1 - ml3$uniquenesses # communalities

##   PINBALL  BILLIARD      GOLF   X.1500M   X.2KROW X.12MINTR     BENCH      CURL
## 0.3649074 0.5863351 0.5445385 0.6388615 0.4804660 0.4621525 0.6980529 0.4641349
##  MAXPUSHU
## 0.4603657
```

```
apply(ml3$loadings^2, 1, sum) ## alternatively

##   PINBALL  BILLIARD      GOLF   X.1500M   X.2KROW X.12MINTR     BENCH      CURL
## 0.3649108 0.5863366 0.5445418 0.6388604 0.4804635 0.4621496 0.6980519 0.4641311
##  MAXPUSHU
## 0.4603640

diag(ml3$loadings %*% t(ml3$loadings)) ## alternatively 2

##   PINBALL  BILLIARD      GOLF   X.1500M   X.2KROW X.12MINTR     BENCH      CURL
## 0.3649108 0.5863366 0.5445418 0.6388604 0.4804635 0.4621496 0.6980519 0.4641311
##  MAXPUSHU
## 0.4603640
```

The table of the output above reports the sum of squared loadings for each factor. For example, $0.4252^2 + 0.4432^2 + \ldots + 0.5122^2 = 1.912$ represents the part of the total variance that is explained by the first factor. If we divide it by the total variance (i.e. 9 in this case since we are working on standardized variables) we obtain the proportion of the total variance explained by the first factor: 21.2%.

The same results can be obtained as follows:

```
total.var <- sum(1 - ml3$uniquenesses) + sum(ml3$uniquenesses)
apply(ml3$loadings^2, 2, sum) / total.var # % of the total.var explained by each fact

##   Factor1   Factor2   Factor3
## 0.2124480 0.1716934 0.1380596
```

The unrotated factors do not have a clear interpretation. In order to improve the understanding of the problem we can try to rotate the axes with the VARIMAX procedure:

```
(ml4 <- factanal(x, factors = 3, rotation = "varimax"))

##
## Call:
## factanal(x = x, factors = 3, rotation = "varimax")
##
## Uniquenesses:
##   PINBALL  BILLIARD      GOLF   X.1500M   X.2KROW X.12MINTR     BENCH      CURL
##     0.635     0.414     0.455     0.361     0.520     0.538     0.302     0.536
##  MAXPUSHU
##     0.540
##
## Loadings:
##           Factor1 Factor2 Factor3
## PINBALL             0.131   0.590
## BILLIARD                    0.765
## GOLF                        0.735
## X.1500M    0.779  -0.179
```

```
## X.2KROW    0.585   0.372
## X.12MINTR  0.678
## BENCH     -0.119   0.816   0.137
## CURL                0.674
## MAXPUSHU   0.433   0.522
##
##                Factor1 Factor2 Factor3
## SS loadings      1.613   1.584   1.502
## Proportion Var   0.179   0.176   0.167
## Cumulative Var   0.179   0.355   0.522
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 12.94 on 12 degrees of freedom.
## The p-value is 0.373
```

As expected from the invariance of the factor model to orthogonal rotations, the estimates of the communalities do not change after rotation.
To facilitate the visualization of the result, we can also "clean up" the factor pattern. We can, for example, define a `cutoff` for the loadings to be printed and reduce the number of digits, e.g. to 2.

```
ml4$loadings # default

##
## Loadings:
##           Factor1 Factor2 Factor3
## PINBALL             0.131   0.590
## BILLIARD                    0.765
## GOLF                        0.735
## X.1500M    0.779  -0.179
## X.2KROW    0.585   0.372
## X.12MINTR  0.678
## BENCH     -0.119   0.816   0.137
## CURL                0.674
## MAXPUSHU   0.433   0.522
##
##                Factor1 Factor2 Factor3
## SS loadings      1.613   1.584   1.502
## Proportion Var   0.179   0.176   0.167
## Cumulative Var   0.179   0.355   0.522

print(ml4$loadings, cutoff = 0.2, digits = 2) # pretty-printed

##
## Loadings:
##           Factor1 Factor2 Factor3
## PINBALL                     0.59
## BILLIARD                    0.76
## GOLF                        0.73
## X.1500M    0.78
```

```
## X.2KROW    0.58    0.37
## X.12MINTR  0.68
## BENCH              0.82
## CURL               0.67
## MAXPUSHU   0.43    0.52
##
##                Factor1 Factor2 Factor3
## SS loadings       1.61    1.58    1.50
## Proportion Var    0.18    0.18    0.17
## Cumulative Var    0.18    0.36    0.52
```

Now it is evident that there are 3 factors. The traditional approach to naming factors is to:

- examine the variables that load heavily on the factor;
- try do decide what construct is common to these variables;
- name the factor after that construct.

The first factor is something that is common to strong performance in a 1500 meter run, a 2000 meter row, and a 12 minute run (a good name for this factor is "Endurance"). By their structure, the other two factors might be named "Strength", and "Hand-Eye Coordination". This seems reasonable looking at correlation matrix.

We may want to calculate an individual's score on the latent variable(s). In factor analysis it is not straightforward, because the factors are random variables which have a probability distribution. There are various methods for obtaining predicted factor scores; the function `factanal` produces scores only if a data matrix is supplied and used. The first method is the regression method of Thompson, the second the weighted least squares method of Bartlett.

```
scores_thompson <- factanal(x, factors = 3, scores = "regression")$scores
scores_bartlett <- factanal(x, factors = 3, scores = "Bartlett")$scores
```

If we were interested in identifying students who have good "strength", we would order observations according to decreasing scores related to the second factor

```
stud.ord <- order(scores_thompson[,2], decreasing = T)
head(stud.ord)

## [1] 415 454 165 397 311 102
```

Student 415 is the one who have the best performances in the disciplines which require strength.

4.9.3 Example 3: intelligence test

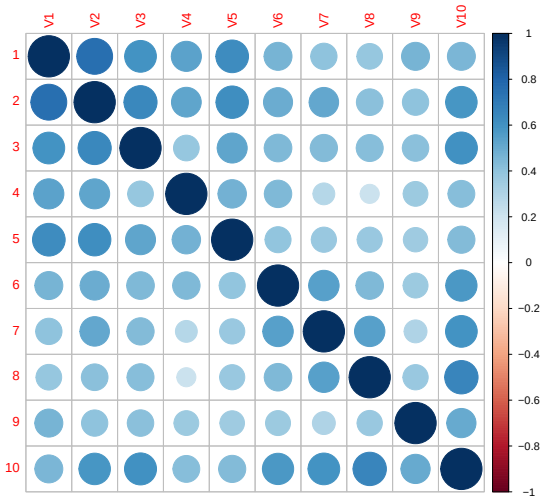
File `intel_test.txt` contains correlations between scores of 75 children in 10 intelligence tests WPPSI (seems the same dataset seen during theoretical lessons?):

- `V1`: information
- `V2`: vocabulary
- `V3`: arithmetic
- `V4`: similarities
- `V5`: comprehension
- `V6`: animal houses
- `V7`: figures completion
- `V8`: labyrinths
- `V9`: geometric design
- `V10`: block design.

```
# intel dataset
if (FALSE) x <- read.table("multivariate_statistics/data/intel_test.txt")
x <- read.table("data/intel_test.txt") # it is a correlation matrix
(R <- as.matrix(x))
```

```
##          V1    V2    V3    V4    V5    V6    V7    V8    V9   V10
##  [1,] 1.000 0.755 0.592 0.532 0.627 0.460 0.407 0.387 0.461 0.459
##  [2,] 0.755 1.000 0.644 0.520 0.617 0.497 0.511 0.417 0.406 0.583
##  [3,] 0.592 0.644 1.000 0.388 0.529 0.449 0.436 0.428 0.412 0.602
##  [4,] 0.532 0.528 0.388 1.000 0.475 0.442 0.280 0.214 0.361 0.424
##  [5,] 0.627 0.617 0.529 0.475 1.000 0.398 0.373 0.372 0.350 0.433
##  [6,] 0.460 0.497 0.449 0.442 0.398 1.000 0.545 0.446 0.366 0.575
##  [7,] 0.407 0.511 0.436 0.280 0.373 0.545 1.000 0.542 0.308 0.590
##  [8,] 0.387 0.417 0.428 0.214 0.372 0.446 0.542 1.000 0.375 0.654
##  [9,] 0.461 0.406 0.412 0.361 0.355 0.366 0.308 0.375 1.000 0.502
## [10,] 0.459 0.583 0.602 0.424 0.433 0.575 0.590 0.654 0.502 1.000
```

```
corrplot::corrplot(R)
```



We can notice a strong correlation between the 10 tests; in fact, all the correlation values are positive and mostly varies between 0.4 and 0.6.
We start performing FA with 2 factors by `factanal` using ML approach

```
(ml1 <- factanal(covmat = R, factors = 2, n.obs = 75, rotation = "none"))
```

```
##
## Call:
## factanal(factors = 2, covmat = R, n.obs = 75, rotation = "none")
##
## Uniquenesses:
##    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10
## 0.215 0.249 0.452 0.622 0.482 0.553 0.534 0.481 0.679 0.177
##
## Loadings:
##       Factor1 Factor2
##  [1,]  0.789  -0.403
##  [2,]  0.834  -0.234
##  [3,]  0.740
##  [4,]  0.587  -0.185
##  [5,]  0.676  -0.247
##  [6,]  0.654   0.140
##  [7,]  0.641   0.235
##  [8,]  0.630   0.351
##  [9,]  0.564
## [10,]  0.807   0.414
##
##                Factor1 Factor2
## SS loadings      4.872   0.685
## Proportion Var   0.487   0.069
## Cumulative Var   0.487   0.556
##
## Test of the hypothesis that 2 factors are sufficient.
```

```
## The chi square statistic is 16.51 on 26 degrees of freedom.
## The p-value is 0.923
```

Since the p-value is very large, we can assume that 2 factors are enough to explain the original correlations.
The percentage of variability of each variable explained by the model is contained in the communalities:

```
apply(res$loadings^2,1,sum)

## Error:  oggetto 'res' non trovato
```

When we don't perform rotation the first factor has high entries for all the variables (its an average measurmente) while the other are contrasts. The VARIMAX rotation works on the factor loadings by letting lower weights converge to zero and higher weights converge to one. Interpretation of the factor loadings is thus facilitated.

```
(ml2 <- factanal(covmat = R, factors = 2, n.obs = 75, rotation = 'varimax'))

##
## Call:
## factanal(factors = 2, covmat = R, n.obs = 75, rotation = "varimax")
##
## Uniquenesses:
##    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10
## 0.215 0.249 0.452 0.622 0.482 0.553 0.534 0.481 0.679 0.177
##
## Loadings:
##       Factor1 Factor2
##  [1,] 0.852   0.245
##  [2,] 0.769   0.399
##  [3,] 0.563   0.481
##  [4,] 0.555   0.266
##  [5,] 0.662   0.281
##  [6,] 0.382   0.549
##  [7,] 0.308   0.609
##  [8,] 0.220   0.686
##  [9,] 0.375   0.424
## [10,] 0.307   0.854
##
##                Factor1 Factor2
## SS loadings      2.904   2.653
## Proportion Var   0.290   0.265
## Cumulative Var   0.290   0.556
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 16.51 on 26 degrees of freedom.
## The p-value is 0.923
```

The first factor exhibits the highest weights on the first five variables and, thus, it can be assumed as a measure of school cleaverness. The second factor, instead, shows large loadings in correspondence of the remaining five variables; by its nature, it discriminates children with marked practical abilities from all the others.

Capitolo 5

Discriminant analysis and supervised classification

The purpose of linear discriminant analysis (LDA) is to find the linear combinations of the original variables that give the best possible separation between the groups in our data set. Linear discriminant analysis is also known as canonical discriminant analysis" or simply "discriminant analysis".

5.1 Discriminant analysis

Methods is based on *linear combinations*: aim is to find coefficients of the linear combination which is best at dividing some known groups. It's an old method, invented by Fisher in 1936, which is still very useful because it's interpretable (if we use a neural network to do the same it all become very less clear).

Suppose we have two population, Π_1 and Π_2 where we can observe a random vector $\mathbf{X}$ $(p \times 1)$ which has different mean ($\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ in the two population) but common variance/covariance $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$ (only assumptions Fisher made was homoscedasticity).
We draw a sample from the two population C_1 and C_2, composed by n_1 units and n_2 units respectively, with mean $\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2$ and covariances S_1, S_2 (say the unbiased version one but it is not necessary).
To separate two groups, something that often does better than original variables is a linear combination: Fisher proposed to look for the linear combination of the observed variables such that, when projected along it, the groups are maximally separated (distant means) and at the same time maximally homogeneous (variance within group is small).
He wanted to find a such as

$$y = \mathbf{a}^\top \mathbf{X}$$

the groups in y are maximally separated and homogeneous.
Fisher developed LDA, consistently with the idea of Anova, to find $\mathbf{a}$ such that when projected, variance of y between group is maximum and variance within group is minimum.
To procede we need to define (in the general case of G groups):

- the mean of y in a single group is the linear combination of the original data means

$$\begin{aligned}\overline{y}_1 &= \mathbf{a}^\top \overline{\mathbf{x}}_1 \\ \overline{y}_2 &= \mathbf{a}^\top \overline{\mathbf{x}}_2 \\ &\dots \\ \overline{y}_G &= \mathbf{a}^\top \overline{\mathbf{x}}_G\end{aligned}$$

- the overall mean is a weighted mean of the two above

$$\overline{y} = \frac{n_1\overline{y}_1 + n_2\overline{y}_2 + \dots n_G\overline{y}_G}{n_1 + n_2 + \dots + n_G} = \frac{n_1\mathbf{a}^\top\overline{\mathbf{x}}_1 + n_2\mathbf{a}^\top\overline{\mathbf{x}}_2 + \dots + n_G\mathbf{a}^\top\overline{\mathbf{x}}_G}{n}$$

5.1.1 Variance between

The *between variance* of y is the between group sum of square divided by the degrees of freedom (here G is the general number of groups, here we have two so $G = 2$)

$$V(y)_{\text{between}} = \frac{BSS}{df} = \frac{\sum_{g=1}^{G}(\overline{y}_g - \overline{y})^2 \cdot n_g}{G-1}$$

the degrees of freedom are due to G means and 1 constraint (1 grand mean): in the two group cases, $df = 1$.
To express in matrix form

$$\begin{aligned}V(y)_{\text{between}} &= \frac{1}{G-1}\sum_{g=1}^{G}(\overline{y}_g - \overline{y})^2 \cdot n_g = \frac{1}{G-1}\sum_{g=1}^{G}(\mathbf{a}^\top\overline{\mathbf{x}}_g - \mathbf{a}^\top\overline{\mathbf{x}})^2 n_g \\ &= \frac{1}{G-1}\sum_{g=1}^{G}\left[\mathbf{a}^\top(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})\right]^2 n_g = \frac{1}{G-1}\sum_{g=1}^{G}\mathbf{a}^\top(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})^T\mathbf{a}^\top n_g \\ &= \frac{1}{G-1}\mathbf{a}^\top\left[\sum_{g=1}^{G}(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})^T \cdot n_g\right]\mathbf{a}^\top\end{aligned}$$

Now if we name the original variables between groups variance

$$\mathbf{B} = \frac{1}{G-1}\sum_{g=1}^{G}\left[(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})^T \cdot n_g\right]$$

where $\mathbf{B}$ is the between group variance in the original X space. We can finally rewrite the variance between of y as function of variance between of $\mathbf{x}$:

$$V(y) = \mathbf{a}^\top\mathbf{B}\mathbf{a}$$

This for the general case.

Important remark 67. Now regarding $\mathbf{B}$:

- dimension of matrix $\mathbf{B}$: with p variables B is a $p \times p$ matrix

- what's its rank? In the general case has at most rank $G-1$

Thus the matrix B **is not full rank**.

Important remark 68. An alternative way to write $\mathbf{B}$, in case we have only two groups is the following

$$\begin{aligned}
V(y) &= \sum_{g=1}^{2}(\overline{y}_g - \overline{y})^2 \cdot n_g = (\overline{y}_1 - \overline{y})^2 \cdot n_1 + (\overline{y}_2 - \overline{y})^2 \cdot n_2 \\
&= n_1\left[\mathbf{a}^\top\overline{\mathbf{x}}_1 - \frac{1}{n}\left(\mathbf{a}^\top\overline{\mathbf{x}}_1 n_1 + \mathbf{a}^\top\overline{\mathbf{x}}_2 n_2\right)\right]^2 + n_2\left[\mathbf{a}^\top\overline{\mathbf{x}}_2 - \frac{1}{n}\left(\mathbf{a}^\top\overline{\mathbf{x}}_1 n_1 + \mathbf{a}^\top\overline{\mathbf{x}}_2 n_2\right)\right]^2 \\
&= \frac{n_1\left[n\mathbf{a}^\top\overline{\mathbf{x}}_1 - \mathbf{a}^\top\overline{\mathbf{x}}_1 n_1 - \mathbf{a}^\top\overline{\mathbf{x}}_2 n_2\right]^2}{n^2} + n_2\frac{\left[n\mathbf{a}^\top\overline{\mathbf{x}}_2 - \mathbf{a}^\top\overline{\mathbf{x}}_1 n_1 - \mathbf{a}^\top\overline{\mathbf{x}}_2 n_2\right]^2}{n^2} \\
&= \frac{n_1}{n^2}\left(\mathbf{a}^\top\overline{\mathbf{x}}_1 n_2 - \mathbf{a}^\top\overline{\mathbf{x}}_2 n_2\right)^2 + \frac{n_2}{n^2}\left(\mathbf{a}^\top\overline{\mathbf{x}}_2 n_1 - \mathbf{a}^\top\overline{\mathbf{x}}_1 n_1\right)^2 \\
&= \frac{n_1 n_2^2}{n^2}\left[\mathbf{a}^\top(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)\right]^2 + \frac{n_2 n_1^2}{n^2}\left[\mathbf{a}^\top(\overline{\mathbf{x}}_2 - \overline{\mathbf{x}}_1)\right]^2 \\
&= \frac{n_1 n_2^2}{n^2}\mathbf{a}^\top(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top\mathbf{a} + \frac{n_1^2 n_2}{n^2}a^\top(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top\mathbf{a} \\
&= \frac{n_1 n_2^2 + n_1^2 n_2}{n^2}\mathbf{a}^\top(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top\mathbf{a} \\
&\overset{(1)}{=} \frac{n_1 \cdot n_2}{n_1 + n_2}\mathbf{a}^\top(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top\mathbf{a}
\end{aligned}$$

where in (1) we simplified using $n^2 = (n_1 + n_2)^2$.
So in the two groups cases here we have that

$$\mathbf{B} = \frac{n_1 \cdot n_2}{n_1 + n_2}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top$$

Remark 76. In the two groups case $\mathbf{B}$ is the product of the col vector times row vector: the product is rank 1 (since both row and col vector has rank 1). So in case of in the case of two group $\mathbf{B}$ has rank 1.

5.1.2 Variance within

Now let's check variance within, that is the sum of variances withing group. Using the unbiased estimator (here below $\mathbf{x}_g$ are the units from group g)

$$\begin{aligned}
V(y)_{\text{within}} &= \frac{\sum_{g=1}^{G}\operatorname{Var}[y_g](n_g - 1)}{n - G} = \frac{\sum_{g=1}^{G}\operatorname{Var}\left[\mathbf{a}^T\mathbf{x}_g\right](n_g - 1)}{n - G} \\
&= \frac{\sum_{g=1}^{G}\mathbf{a}^T\operatorname{Var}[\mathbf{x}_g]\mathbf{a}\cdot(n_g - 1)}{n - G} \overset{(1)}{=} \frac{\sum_{g=1}^{G}\mathbf{a}^T\mathbf{S}_g\mathbf{a}\cdot(n_g - 1)}{n - G} \\
&= \mathbf{a}^T\left[\frac{\sum_{g=1}^{G}\mathbf{S}_g(n_g - 1)}{n - G}\right]\mathbf{a} \\
&= \mathbf{a}^T\mathbf{W}\mathbf{a}
\end{aligned}$$

where in (1) we used $\operatorname{Var}[aX] = a^2\operatorname{Var}[X]$ and $\mathbf{W}$ is the within group sum of square for x.

Remark 77. Now we rephrased within and between variance in terms of $\mathbf{a}$ so we can choose $\mathbf{a}$ optimizing.

Important remark 69 (Recap). Only assumption so far is that the G population shares covariance matrix (not any hypothesis on shape of distribution) and ended finding the variances of linear combination with $\mathbf{a}$, which are

$$V(Y)_b = \mathbf{a}^\top \mathbf{B}\mathbf{a}$$
$$V(Y)_w = \mathbf{a}^\top \mathbf{W}\mathbf{a}$$

with $\mathbf{B}$ and $\mathbf{W}$ the between and within group variance in the x space. Now we have what needed to derive Fisher criterion.

5.1.3 Fisher's criterion

Fisher wanted to find a linear combination $y = \mathbf{a}^\top X$ such that the ratio

$$\phi = \frac{V(y)_b}{V(y)_w} = \frac{\mathbf{a}^\top \mathbf{B}\mathbf{a}}{\mathbf{a}^\top \mathbf{W}\mathbf{a}}$$

is maximum with respect to $\mathbf{a}$; this means a combination which creates groups that are mostly separated (max variance at the numerator num) and homogeneous (min variance denominator). So our problem is $\max_{\mathbf{a}} \phi$, a standard maximum problem; we derive wrt to $\mathbf{a}$ and equate to 0

$$\frac{\partial \phi}{\partial \mathbf{a}} = \frac{2\mathbf{B}\mathbf{a}(\mathbf{a}^\top \mathbf{W}\mathbf{a}) - 2\mathbf{W}\mathbf{a}(\mathbf{a}^\top \mathbf{B}\mathbf{a})}{(\mathbf{a}^\top \mathbf{W}\mathbf{a})^2} = 2\left[\frac{\mathbf{B}\mathbf{a}(\mathbf{a}^\top \mathbf{W}\mathbf{a})}{(\mathbf{a}^\top \mathbf{W}\mathbf{a})^2} - \frac{\mathbf{W}\mathbf{a}(\mathbf{a}^\top \mathbf{B}\mathbf{a})}{(\mathbf{a}^\top \mathbf{W}\mathbf{a})^2}\right]$$
$$\overset{(1)}{=} 2\left[\frac{\mathbf{B}\mathbf{a}}{\mathbf{a}^\top \mathbf{W}\mathbf{a}} - \frac{\mathbf{W}\mathbf{a}\phi}{\mathbf{a}^\top \mathbf{W}\mathbf{a}}\right]$$

where in (1) we just replaced $\frac{a^\top Ba}{a^\top Wa} = \phi$. Furthermore $\mathbf{a}^\top \mathbf{W}\mathbf{a}$ is a constant $\neq 0$; by equating to 0 for maximization we can simplify to

$$\mathbf{B}\mathbf{a} - \phi\mathbf{W}\mathbf{a} = \mathbf{0}$$
$$(\mathbf{B} - \phi\mathbf{W})\mathbf{a} = \mathbf{0}$$

Now we premultiply both sides by $\mathbf{W}^{-1}$ (under the assumption that $\mathbf{W}$ it is non singular)

$$\left(\mathbf{W}^{-1}\mathbf{B} - \phi\mathbf{I}\right)\mathbf{a} = \mathbf{0}$$

This last is a linear equation system which admits a nontrivial solution iff the determinant

$$\det\left(\mathbf{W}^{-1}\mathbf{B} - \phi\mathbf{I}\right) = 0$$

that is if ϕ is a root of the characteristics polynomial of $\mathbf{W}^{-1}\mathbf{B}$. This means that ϕ is an eigenvalue of $\mathbf{W}^{-1}\mathbf{B}$ and $\mathbf{a}$ is the corresponding eigenvector. If we prefer we can rewrite as the eigenvalue eigenvector relationship

$$(\mathbf{W}^{-1}\mathbf{B} - \phi\mathbf{I})\mathbf{a} = \mathbf{0} \implies \mathbf{W}^{-1}\mathbf{B}\mathbf{a} = \phi\mathbf{a}$$

Important remark 70. How many non zero eigenvalue do we have? the rank of $\mathbf{W}^{-1}\mathbf{B}$

- in the two group cases: the rank of $\mathbf{W}$ is p, while $\mathbf{B}$ is $p \times p$ matrix but its rank is 1. So the product $\mathbf{W}^{-1}\mathbf{B}$ will have rank 1: therefore it will have just 1 non zero eigenvalue.
  In the 2 group cases the **best linear discriminant direction** will be the *eigenvector* of $\mathbf{W}^{-1}\mathbf{B}$ corresponding to the non zero eigenvalue

- in the general G group case: the rank $\mathbf{W}^{-1}\mathbf{B}$ will be at most $G-1$, so will have at most $G-1$ non null eigenvalue and at most $G-1$ discriminant directions (indipendently from how large is p eg I can have 100 variables). The optimal linear discriminant direction will be defined by the *eigenvector* of $W^{-1}B$ corresponding to the *largest eigenvalue* as ϕ.
  In this case we can consider more than 1 discriminant directions: we can consider the *eigenvector corresponding to different eigenvalues in decreasing order* (each of them will have a decreasing discrimination power). [1].
  These eigenvector

  - define a vector subspace containing te variability between features;
  - are primarily used in feature reduction and can be interpreted in the same way as principal compoennts

Remark 78. Fisher proved that in the two group case, the eigenvector of $\mathbf{W}^{-1}\mathbf{B}$ corresponding to the only non zero eigenvalues can be obtained in closed form as

$$\mathbf{a} = \mathbf{W}^{-1}(\overline{x}_1 - \overline{x}_2)$$

5.1.4 Differences with PCA

Important remark 71. So we're again trying to solve a linear equations system as done for principal components. The differences with Principal components:

1. here we don't need any constraint (there we needed to constraint $\mathbf{a}$ to have norm 1, to make fair comparison) because here $\mathbf{a}$ it appears both at numerator and denominator of the ratio, in similar quadratic form so it cancels out (even if $\mathbf{a}$ is long)

2. while the covariance matrix $\mathbf{S}, \mathbf{\Sigma}$ is symmetric, this matrix $\mathbf{W}^{-1}\mathbf{B}$ is non-symmetric (it's the produt of two symmetric matrices but it is not). Spectral decomposition exists but there can be immaginary eigenvalue/vectors

3. it can be that LDA outperform PCA in classification: discriminatory information is not necessary aligned with the direction of maximum variability

[1] eg if we have three groups we can go on and find the second discriminant direction corresponding to the second highest larger eigenvalue

5.1.5 Linear discriminant functions

The linear combinations:

$$y_1 = \mathbf{a}_1^\top \mathbf{x}$$
$$y_2 = \mathbf{a}_2^\top \mathbf{x}$$
$$\dots$$
$$y_{G-1} = \mathbf{a}_{G-1}^\top \mathbf{x}$$

with $\mathbf{a}_1, \dots, \mathbf{a}_{G-1}$ the eigenvectors corresponding to the non zero eigenvalues in decreasing order, are called linear *discriminant functions* (or also *canonical variates*).

5.1.6 Correlation of eigenvectors

Lets' consider two different discriminant directions $\mathbf{a}_i, \mathbf{a}_j$ and their corresponding eigengector ϕ_i, ϕ_j. Being solution we have that hold:

$$\mathbf{B}\mathbf{a}_i = \phi_i \mathbf{W}\mathbf{a}_i$$
$$\mathbf{B}\mathbf{a}_j = \phi_j \mathbf{W}\mathbf{a}_j$$

Let's pre multiply the first by $\mathbf{a}_j^\top$ and the second by $\mathbf{a}_i^\top$

$$\mathbf{a}_j^\top \mathbf{B}\mathbf{a}_i = \phi_i \mathbf{a}_j^\top \mathbf{W}\mathbf{a}_i$$
$$\mathbf{a}_i^\top \mathbf{B}\mathbf{a}_j = \phi_j \mathbf{a}_i^\top \mathbf{W}\mathbf{a}_j$$

On the left hand side of the two equations we have two quantities that are the same: two scalar which is one the transpose of the other (transpose of scalar is the scalar itself), that is

$$\mathbf{a}_j^\top \mathbf{B}\mathbf{a}_i = \mathbf{a}_i^\top \mathbf{B}\mathbf{a}_j$$

thus event the right hand side has to be the same as well

$$\phi_i \mathbf{a}_j^\top \mathbf{W}\mathbf{a}_i = \phi_j \mathbf{a}_i^\top \mathbf{W}\mathbf{a}_j$$

and we have that even $\mathbf{a}_j^\top \mathbf{W}\mathbf{a}_i = \mathbf{a}_i^\top \mathbf{W}\mathbf{a}_j$ for the same reason before (a scalar and its equivalent transpose). Now:

- Given that ϕ_i and ϕ_j are different from zero, and different each other, in order for the equality to hold it must be that

  $$\mathbf{a}_j^\top \mathbf{W}\mathbf{a}_i = \mathbf{a}_i^\top \mathbf{W}\mathbf{a}_j = 0$$

  Now if we put all the discriminant direction together in the matrix $\mathbf{A}$ we have that

  $$\mathbf{A}^\top \mathbf{W}\mathbf{A} \quad \text{is diagonal}$$

  because crossproducts have to be all equal to 0.
  This means that **discriminant directions are uncorrelated within group**: as $\mathbf{a}_i^\top \mathbf{W}\mathbf{a}_j$ is the covariance between y_i and y_j within group (there's $\mathbf{W}$), then we conclude that the linear discriminant functions are uncorrelated within groups.

- having said that

$$\mathbf{a}_i^\top \mathbf{W}\mathbf{a}_j = 0 \implies \mathbf{a}_i^\top \mathbf{B}\mathbf{a}_j = 0$$

  so it means that the discriminant function are **uncorrelated also between groups**.

- If they are uncorrelated both within and between groups, they are overall uncorrelated, with respect to the whole set of units, so this means that $\mathbf{A}^\top \mathbf{S}\mathbf{A}$ is diagonal.
  This is one important difference we'have not asked for: the orthogonality.

Remark 79. Many softwares (R included) scale $\mathbf{A}$ so that $\mathbf{A}^\top \mathbf{W}\mathbf{A}$ not only is diagonal but $\mathbf{A}^\top \mathbf{W}\mathbf{A} = \mathbf{I}$ (the discriminant variables are uncorrelated and have unit within group variance; they're often said to be sphered

5.1.7 Example: iris

Example 5.1.1 (Fisher's Iris Data)**.** We have data on 150 iris flowers belonging to three species (setosa, versicolor and verginica); for each flower the length and width of the flower and sepal length and width. The group means are

$$\begin{aligned}
\overline{\mathbf{x}}_1^T &= \begin{bmatrix} 5.006 & 3.418 & 1.464 & 0.244 \end{bmatrix} \\
\overline{\mathbf{x}}_2^T &= \begin{bmatrix} 5.936 & 2.77 & 4.26 & 1.326 \end{bmatrix} \\
\overline{\mathbf{x}}_3^T &= \begin{bmatrix} 6.558 & 2.974 & 5.552 & 2.026 \end{bmatrix}
\end{aligned}$$

we have three groups, we expect to find two discriminant directions, which respectively correspond to eigenvalue $\phi_1 = 32.27$ and $\phi_2 = 0.278$

$$\begin{aligned}
\mathbf{a}_1^T &= \begin{bmatrix} -0.2048 & 0.3871 & -0.5465 & -0.7138 \end{bmatrix} \\
\mathbf{a}_2^T &= \begin{bmatrix} -0.008 & -0.589 & 0.2543 & -0.767 \end{bmatrix}
\end{aligned}$$

The obtained projected of the first LD for the three groups are respectively $\overline{\mathbf{y}}_{11} = 1.37$ (setosa) $\overline{\mathbf{y}}_{21} = -0.98$ (virsicolor) $\overline{\mathbf{y}}_{31} = -1.98$ (virginica)

- looking at the variables: the first direction separates flower with big sepal and low petal and viceversa

- the second direction $\mathbf{a}_2^\top$ is focused on sepal only

The projection on LD1 and LD2: setosa is most separated on the first LD1 while on the second there's not too much separation. Separation on first is good

5.2 Performing classification

5.2.1 Using LDA

Remark 80. Even if it has been derived for discrimination purposes, Fisher's linear function can also be used to address classification issues, i.e. to define a rule for assigning a unit, whose group membership is unknown, to one out of the G groups.
The method is general, but for teaching purposes we will limit our attention to the two group case only.

In the two groups cases: I have data $\mathbf{x}_0$ on a new unit (a $p \times 1$ vector) whose membership is unknown. Data includes the same variables I have used to obtain the discriminant function. I can calculate

$$y_0 = \mathbf{a}^\top \mathbf{x}_0$$

where $\mathbf{a}$ is the discriminant direction. In the two group case I also compute the projection of the mean of groups $(\overline{\mathbf{x}}_1, \overline{\mathbf{x}}_2)$ along $\mathbf{a}$

$$\overline{y}_2 = \mathbf{a}^\top \overline{\mathbf{x}}_2 \qquad\qquad \overline{y}_1 = \mathbf{a}^\top \overline{\mathbf{x}}_1$$

Assuming for simplicity that $\overline{y}_1 > \overline{y}_2$, to classify my unit I project it on the discriminant direction (calculating y_0) and check the distance between the projection and the means projected

$$\begin{aligned} |y_0 - \overline{y}_2| < |y_0 - \overline{y}_1| &\implies \mathbf{x}_0 \in \Pi_2 \\ |y_0 - \overline{y}_1| < |y_0 - \overline{y}_2| &\implies \mathbf{x}_0 \in \Pi_1 \end{aligned}$$

Alternatively said:

$$\begin{cases} y_0 > \frac{\overline{y}_1 + \overline{y}_2}{2} &\implies \mathbf{x}_0 \in \Pi_1 \\ y_0 < \frac{\overline{y}_1 + \overline{y}_2}{2} &\implies \mathbf{x}_0 \in \Pi_2 \\ y_0 = \frac{\overline{y}_1 + \overline{y}_2}{2} &\implies \text{toss a coin to decide which group} \end{cases}$$

In the two group case, as said previously, Fisher proved that the eigenvector of $\mathbf{W}^{-1}\mathbf{B}$ corresponding to the only direction $\mathbf{a}$ be obtained in closed form as:

$$\mathbf{a} = \mathbf{W}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)$$

By exploiting this we can say

$$y_0 = \mathbf{a}^\top \mathbf{x}_0 = (\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1} \mathbf{x}_0$$

where being $\mathbf{W}$ symmetric, it's transpose coincides with the inverse. The group transformed means become

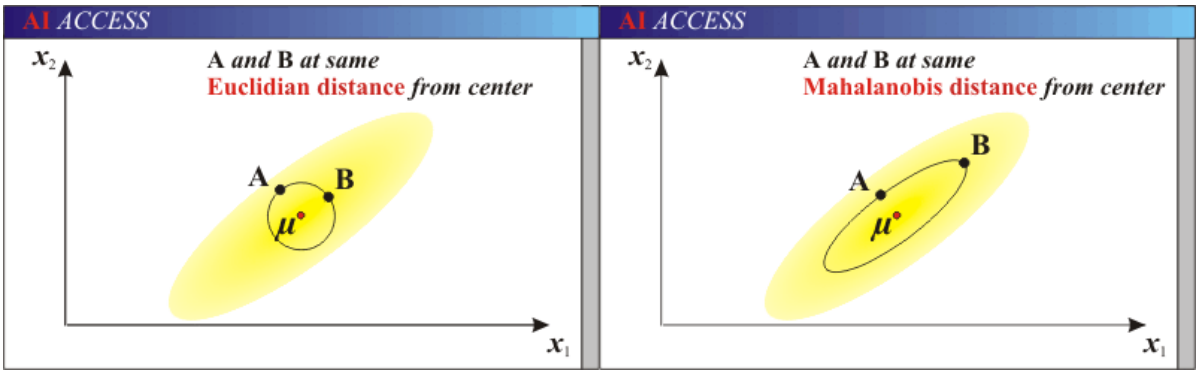
$$\begin{aligned} \overline{y}_1 &= \mathbf{a}^\top \overline{\mathbf{x}}_1 = (\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1} \overline{\mathbf{x}}_1 \\ \overline{y}_2 &= \mathbf{a}^\top \overline{\mathbf{x}}_2 = (\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1} \overline{\mathbf{x}}_2 \end{aligned}$$

So the rule to assign $\mathbf{x}_0$ in Π_1, becomes

$$\begin{aligned} y_0 &> \frac{\overline{\mathbf{y}}_1 + \overline{\mathbf{y}}_2}{2} \\ (\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1} \mathbf{x}_0 &> \frac{1}{2}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1}(\overline{\mathbf{x}}_1 + \overline{\mathbf{x}}_2) \end{aligned} \tag{5.1}$$

This last is called **Fisher 's linear discriminant** classification rule.

Remark 81. This allocation rule is very popular as it can be obtained addressing the classification problem according to a variety of different perspectives. We will see a couple of them in the following.



(a) (b)

Figura 5.1: Euclidean and Mahalanobis distance

5.2.2 Using Mahalanobis distance

Remark 82. Classification problem can be addressed directly in the original p-dimensional space by assigning a unit (whose group membership is unknown) to the population from which it has the smallest Mahalanobis distance (from the mean).

5.2.2.1 The Mahalanobis distance

One common way to measure the distance between points in a p dimensional space is provided by the **Euclidean distance**; however it attaches equal weight to all the axes of the representation.
If differences are present in the variances of the observed variables, and if the variables are correlated this might not be a desirable feature.
In Fig.5.1 a bivariate point cloud is represented: The variables have different variances and are correlated. The points A and B have the same Euclidean distance from the center of the point cloud (i.e. from the average vector) but while B lies in the core of the distribution, A is in a low frequency region.

Mahalanobis distance provides a way to take into account variances and correlations when computing distances, i.e. to take into account the shape of the point cloud. Given a point whose coordinate vector is $\mathbf{x}$, its Mahalonobis distance from the center of the point cloud $\overline{\mathbf{x}}$ is defined as

$$d_M(\mathbf{x}, \overline{\mathbf{x}}) = \sqrt{(\mathbf{x} - \overline{\mathbf{x}})^\top \mathbf{S}^{-1} (\mathbf{x} - \overline{\mathbf{x}})}$$

where as usual, $\mathbf{S}$ is the sample covariance matrix. Mahalanobis distance is thus a weighted Euclidean distance. In Fig.5.1 the same point cloud and two points having the same Mahalanobis distance are represented.

Important remark 72. Mahalanobis is a special kind, actually weighted, of euclidean distance (which consider the covariance/correlation of two variables). This can be seen by the fact that if we substitute $\mathbf{S}^{-1}$ with $\mathbf{I}$ we obtain the euclidean distance: actually Mahalanobis is a weighted euclidean distance, has an inverse covariance matrix in between.

5.2.2.2 Using it for classification

Mahalanobis distance turns out to be really useful also for classification purposes. Assuming homoscedasticity, and estimating the unknown common covariance matrix by the within group covariance matrix $\mathbf{W}$, the *squared* distances between a unit $\mathbf{x}_0$ and groups means:

$$(\mathbf{x}_0 - \overline{\mathbf{x}}_1)^\top \mathbf{W}^{-1}(\mathbf{x}_0 - \overline{\mathbf{x}}_1)$$
$$(\mathbf{x}_0 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1}(\mathbf{x}_0 - \overline{\mathbf{x}}_2)$$

So the rule to assign to Π_1 becomes

$$(\mathbf{x}_0 - \overline{\mathbf{x}}_1)^\top \mathbf{W}^{-1}(\mathbf{x}_0 - \overline{\mathbf{x}}_1) < (\mathbf{x}_0 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1}(\mathbf{x}_0 - \overline{\mathbf{x}}_2)$$
$$\mathbf{x}_0^\top \mathbf{W}^{-1}\mathbf{x}_0 - \overline{\mathbf{x}}_1^\top \mathbf{W}^{-1}\mathbf{x}_0 - \mathbf{x}_0^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_1 + \overline{\mathbf{x}}_1^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_1 < \mathbf{x}_0^\top \mathbf{W}^{-1}\mathbf{x}_0 - \mathbf{x}_2^\top \mathbf{W}^{-1}\mathbf{x}_0 - \mathbf{x}_0^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_2 + \overline{\mathbf{x}}_2^\top \mathbf{W}^{-1}$$

now

- we simplify the first two terms ($\mathbf{x}_0^\top \mathbf{W}^{-1}\mathbf{x}_0$) in both members of inequality
- note that $\overline{\mathbf{x}}_1^\top \mathbf{W}^{-1}\mathbf{x}_0$ is a constant, transpose of $\mathbf{x}_0^T \mathbf{W}^{-1}\overline{\mathbf{x}}_1$, so coincides with it and we gather/put together
- same happens for $\overline{\mathbf{x}}_2^\top \mathbf{W}^{-1}\mathbf{x}_0$ and $\mathbf{x}_0^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_2$ are

Thus:

$$-2\overline{\mathbf{x}}_1 \mathbf{W}^{-1}\mathbf{x}_0 + \overline{\mathbf{x}}_1^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_1 < -2\overline{\mathbf{x}}_2 \mathbf{W}^{-1}\mathbf{x}_0 + \overline{\mathbf{x}}_2^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_2$$
$$2\overline{\mathbf{x}}_1^\top \mathbf{W}^{-1}\mathbf{x}_0 - 2\overline{\mathbf{x}}_2 \mathbf{W}^{-1}\mathbf{x}_0 > \overline{\mathbf{x}}_1^\top \mathbf{W}^{-1}\overline{\mathbf{x}}_1 - \mathbf{x}_2 \mathbf{W}^{-1}\overline{\mathbf{x}}_2$$

at the second member we have a difference of two squares. By gathering, we end with the same rule as Fisher's one:

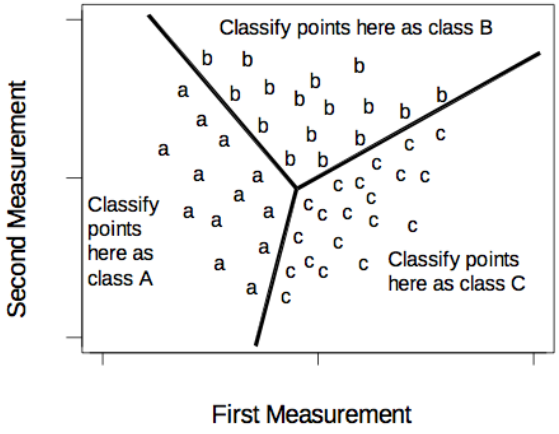
$$(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1}\mathbf{x}_0 > \frac{1}{2}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top \mathbf{W}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2) \tag{5.2}$$

Important remark 73. So assigning a unit to the population it is closest to in the linear discriminant space is the same as assigning the unit to the populationo which it has the smallest Mahalanobis distance in the original p-dimensional space.

Remark 83. Ine final comment: in the linear discriminant space the linear discriminant function are uncorrelated (so we don't need a weight that accounts for correlation).
Thus using Mahalanobis is original space is same as using euclidean in the transformed space.

Important remark 74. The method can be extended to the classification of units into more than 2 groups (5.2). In the G group case we allocate an object to the group for which either

- (a) the Mahalanobis distance between the object and the class mean is smallest using the original variables or
- (b) the Euclidean distance between the object and the class mean is in the subspace spanned by the Canonical Variates



(a)

Figura 5.2: Classification with three groups and mahalanobis distance

5.2.3 Classification based on probability models

So far we have addressed classification issues in a purely distribution free context. Now we will assume that we know the shape of the probability density functions (pdf) that have generated our data in the G groups. We will consider the $G = 2$ case only.
Let:

- $\mathbf{x}$ be the p-dimensional vector of the observed variables
- we call R the sample space: set of all the possible values that $\mathbf{x}$ can take;
- $\mathbf{x}_0$ a new unit whose group membership is unknown
- Π_1, Π_2 the two populations to which either it can belong
- we assume to know the density of the two populations

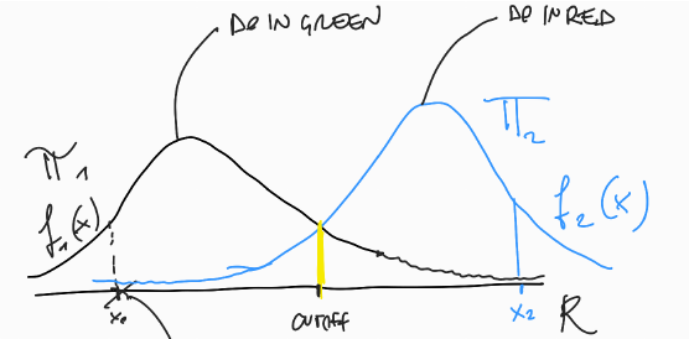
$$f(\mathbf{x}|\mathbf{x} \in \Pi_1) = f_1(\mathbf{x})$$
$$f(\mathbf{x}|\mathbf{x} \in \Pi_2) = f_2(\mathbf{x})$$

The key assumption is that x has a different pdf in Π_1 and Π_2. As $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ usually overlap, each point of R can belong both to Π_1 and Π_2, but with a different probability degree

The goal is to partition R into two regions R_1 and R_2 (mutually exclusive and exaustive, that is $R_1 \cup R_2 = R$ and $R_1 \cap R_2 = \emptyset$) such that that

- if $\mathbf{x}_0$ falls in R_1 it is assigned to population Π_1
- if $\mathbf{x}_0$ falls in R_2 it is assigned to Π_2 and
- probability of a wrong assignment is minimized

Once found the split in R_1 and R_2 it can be used to classify.



(a)

Figura 5.3: Two densities

5.2.3.1 Two population with same prior probability

assume we've two population with densities as depicted in figure 5.3: populations are not perfectly separated, tails overlap, the two population shares a part of the domain. We want to split the domain R:

- in case of the $\mathbf{x}_0$ on the figure it is more likely that the unit comes from Π_1 than Π_2. so I assign it to Π_1
- in case of $\mathbf{x}_2$ it is more liklely that the unit comes from Π_2 than from Π_1, so I assign it to Π_2
- the cutoff between the two population will be at the cross of the two curves in yellow. the split is with R_1 to the left and R_2 on the right

All above far can be translated in a very simple classification rule:

$$\frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} > 1 \implies \mathbf{x}_0 \in \Pi_1$$

because it's more likely that $\mathbf{x}_0$ comes from population 1 than if it comes from population 2. The partition (and classifier associated) would be:

$$R_1 = \left\{ \mathbf{x}_0 : \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} > 1 \right\}$$
$$R_2 = \left\{ \mathbf{x}_0 : \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} < 1 \right\}$$
$$\text{toss a coin if } \left\{ x : \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} = 1 \right\}$$

So finally, assuming I know the distributions f_1, f_2, I can compute the likelihood of the data, and then the ratio of likelihood and finally choose.
This rule is **likelihood ratio rule** (f_1 and f_2 plays the role of likelihood).

5.2.3.2 Two population with different prior probability

Things are not always so simple: it can be that not all population are equally likely have the same a priori probability (eg rare disease common disease we need to take in account). Now let:

- π_1 to be the prior probability that a unit comes from population 1
- π_2 to be the prior probability that a unit comes from population 2
- $\pi_1 + \pi_2 = 1$

The total probability of a wrong classification p is:

$$p = p(1|2) + p(2|1)$$

that is the sum of probability that I assign a unit to population 1 when it comes from population 2 and the viceversa. This two kind of mistake could have different probabilities which are

$$p(2|1) = \pi_1 \int_{R_2} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x}$$
$$p(1|2) = \pi_2 \int_{R_1} f_2(\mathbf{x})\ \mathrm{d}\mathbf{x}$$

where, eg $\int_{R_2} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x}$ is the probability of being wrongly classified in population 2 for those who belong to population 1. Thus the total probability of a wrong classification becomes

$$p = \pi_1 \int_{R_2} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x} + \pi_2 \int_{R_1} f_2(\mathbf{x})\ \mathrm{d}\mathbf{x}$$

We want to find R_1 and R_2 so that this quantity is minimum. To find the solution we need to remember that

$$\int_R f_1(\mathbf{x})\ \mathrm{d}\mathbf{x} = 1,$$
$$\int_R f_2(\mathbf{x})\ \mathrm{d}\mathbf{x} = 1,$$
$$R_1 \cup R_2 = R$$
$$R_1 \cap R_2 = \emptyset$$

Thus considering the first density can be split in the two areas

$$\int_R f_1(\mathbf{x})\ \mathrm{d}\mathbf{x} = \int_{R_1} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x} + \int_{R_2} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x} = 1$$

So the quantity i'm interested in (the probability of wrong Π_2 classification of units belonging to Π_1) is

$$\int_{R_2} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x} = 1 - \int_{R_1} f_1(\mathbf{x})\ \mathrm{d}\mathbf{x}$$

Going back to the total probability of wrong classification we have:

$$\begin{aligned} p &= \pi_1 \int_{R_2} f_1(\mathbf{x})\, \mathrm{d}\mathbf{x} + \pi_2 \int_{R_1} f_2(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \pi_1 \left[1 - \int_{R_1} f_1(\mathbf{x})\, \mathrm{d}\mathbf{x}\right] + \pi_2 \int_{R_1} f_2(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \pi_1 - \pi_1 \int_{R_1} f_1(\mathbf{x})\, \mathrm{d}\mathbf{x} + \pi_2 \int_{R_1} f_2(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \pi_1 + \int_{R_1} \pi_2 f_2(\mathbf{x}) - \pi_1 f_1(\mathbf{x})\, \mathrm{d}\mathbf{x} \end{aligned}$$

To minimize this, we have to chose R_1 such that the integral is minimum. And this occurs for values of $\mathbf{x}$ for which the integrand is negative

$$\begin{aligned} R_1 &= \{x : \pi_2 f_2(\mathbf{x}) - \pi_1 f_1(\mathbf{x}) < 0\} \\ &= \{x : \pi_1 f_1(\mathbf{x}) > \pi_2 f_2(\mathbf{x})\} \\ &= \left\{x : \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} > \frac{\pi_2}{\pi_1}\right\} \end{aligned}$$

In the last one the first inequality member is a likelihood ratio as seen before and the second the switched ratio on priors probability.
So similarly to what seen before the allocation rule will become

$$\begin{aligned} \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} &> \frac{\pi_2}{\pi_1} \implies \mathbf{x}_0 \in \Pi_1 \\ \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} &< \frac{\pi_2}{\pi_1} \implies \mathbf{x}_0 \in \Pi_2 \\ \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} &> \frac{\pi_2}{\pi_1} \implies \text{toss a coin to allocate} \end{aligned}$$

Important remark 75. This is the rule minimizing the total probability of a wrong classification in case of different prior probabilities: likelihood ratio are compared with inverse priors ratio. If $\pi_1 = \pi_2 = 1/2$, then the rule minimizing the total probability of a wrong classification coincides with the likelihood ratio rule seen before, otherwise results are different and take into account the fact theat different population have different prior probabilities.

Important remark 76. It is worth adding that the classification rule obtained by minimizing the total probability of a wrong classification is equivalent to the one that would be obtained by maximizing the posterior probability of population membership. That's the reason why it is often called an optimal Bayes rule.

5.2.3.3 Using Bayes Theorem

The same result above can be obtained using Bayes theorem. For the following:

- $\mathbb{P}(X \in \Pi_1 | X = \mathbf{x}_0)$ is the posterior probability for a observed unit to belong to population Π_1
- $\mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1) = f(X = \mathbf{x}_0 | X \in \Pi_1)$ is the likelihood of a single observation

we have that the posterior probability for population 1 and 2 can be rewritten according to Bayes rule as

$$\mathbb{P}(X \in \Pi_1 | X = \mathbf{x}_0) = \frac{\mathbb{P}(X \in \Pi_1) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1)}{\mathbb{P}(X \in \Pi_1) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1) + \mathbb{P}(X \in \Pi_2) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)}$$
$$\mathbb{P}(X \in \Pi_2 | X = \mathbf{x}_0) = \frac{\mathbb{P}(X \in \Pi_2) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)}{\mathbb{P}(X \in \Pi_1) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1) + \mathbb{P}(X \in \Pi_2) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)}$$
$$= 1 - \mathbb{P}(X \in \Pi_1 | X = \mathbf{x}_0)$$

I can assign a unint to a population which have the largest posterior probability of coming from. The ensuing rule is called the optimal bayes's rule:

$$\frac{\mathbb{P}(X \in \Pi_1) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1)}{\mathbb{P}(X \in \Pi_1) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1) + \mathbb{P}(X \in \Pi_2) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)} > \dots$$
$$\dots \frac{\mathbb{P}(X \in \Pi_2) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)}{\mathbb{P}(X \in \Pi_1) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_1) + \mathbb{P}(X \in \Pi_2) \cdot \mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)}$$

To simplify notation substitute the priors, eg $\mathbb{P}(X \in \Pi_1)$ with π_1 and likelihoods, eg $\mathbb{P}(X = \mathbf{x}_0 | X \in \Pi_2)$ with $f_2(\mathbf{x}_0)$. The rule becomes:

$$\frac{\pi_1 f_1(\mathbf{x}_0)}{\pi_1 f_1(\mathbf{x}_0) + \pi_2 f_2(\mathbf{x}_0)} > \frac{\pi_2 f_2(\mathbf{x}_0)}{\pi_1 f_1(\mathbf{x}_0) + \pi_2 f_2(\mathbf{x}_0)}$$
$$\pi_1 f_1(\mathbf{x}_0) > \pi_2 f_2(\mathbf{x}_0)$$
$$\frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} > \frac{\pi_2}{\pi_1}$$

which ends to be the same rule found before: if above holds we assign to Π_1 otherwise to Π_2. The classification rule minimizing the total probability of a wrong classification is the optimal Bayes rule, that is, it assign a unit to the population it has the largest posterior probability of coming from

$$R_1 = \left\{ x : \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} > \frac{\pi_2}{\pi_1} \right\}$$
$$R_2 = \left\{ x : \frac{f_1(\mathbf{x}_0)}{f_2(\mathbf{x}_0)} < \frac{\pi_2}{\pi_1} \right\}$$

this is the rule that we will use most of the cases.

5.2.3.4 Obtaining the likelihoods

Now the problem is still we said nothing on how to obtain the multivariate likelihoods (f_1 and f_2) for the last two coinciding rules (likelihood ratio and bayesian).

One way to do it is to define a model for $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ and then need to estimate the parameters. If $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are multivariate normal distributions we have that the densities (il valore assoluto per indicare il determinante)

$$f_1(\mathbf{x}) = (2\pi)^{-p/2} |\mathbf{\Sigma}_1|^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)^\top \mathbf{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) \right\}$$
$$f_2(\mathbf{x}) = (2\pi)^{-p/2} |\mathbf{\Sigma}_2|^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)^\top \mathbf{\Sigma}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right\}$$

Then the likelihoods ratio becomes

$$\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \ldots = |\boldsymbol{\Sigma}_1|^{-1/2} |\boldsymbol{\Sigma}_2|^{-1/2} \cdot \ldots$$
$$\ldots \cdot \exp\left\{-\frac{1}{2}\left[\mathbf{x}^\top(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2)\mathbf{x} - 2\mathbf{x}^\top(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) + \boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right]\right\}$$

People usually log this stuff above, obtaining

$$\begin{aligned}\log\left(\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}\right) &= \frac{1}{2}\log\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right)\ldots \\ &- \frac{1}{2}\left[\mathbf{x}^\top\left(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}\right)\mathbf{x} - 2\mathbf{x}^\top(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) + \boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\right] \\ &= Q(\mathbf{x})\end{aligned}$$

$Q(\mathbf{x})$ is so called quadratic discriminant, a function for heteroschedastic normal population; with two mnv with different variances, the optimal surface that separate them is a quadratic surface (a parabola).
We need to compare $Q(\mathbf{x})$ with

- 0 (log of 1 where the two f are equivalent)
- $\log(\pi_2/\pi_1)$ if using populations with different prior

In case of homoscedasticity $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$ the likelihood ratio simplifies

$$\begin{aligned}\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} &= |\boldsymbol{\Sigma}|^{-1/2} |\boldsymbol{\Sigma}|^{1/2} \cdot \exp\left\{-\frac{1}{2}\left[\mathbf{x}^\top(\boldsymbol{\Sigma} - \boldsymbol{\Sigma})\mathbf{x} - 2\mathbf{x}^\top(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2) + \boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right]\right\} \\ &= \exp\left\{-\frac{1}{2}\left[-2\mathbf{x}^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right]\right\} \\ &= \exp\left[\mathbf{x}^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right]\end{aligned}$$

Taking the log we obtain

$$\begin{aligned}\log\left(\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}\right) &= \mathbf{x}^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ &\overset{(1)}{=} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\end{aligned}$$

where in (1) is a scalar so i can transpose it obtaining the same. The final equation has to be compared to 0 or $\log(\pi_2/\pi_1)$ as usual.

This above is at the population level. At the sample level we estimate $\boldsymbol{\mu}_1$ with $\overline{\mathbf{x}}_1$ and $\boldsymbol{\mu}_2$ with $\overline{\mathbf{x}}_2$ and $\boldsymbol{\Sigma}$ with $\mathbf{W}$ and considering obtaining equal priors probability (thus comparing to 0) we have:

$$(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top\mathbf{W}^{-1}\mathbf{x} > \frac{1}{2}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)^\top\mathbf{W}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)$$

Important remark 77. In it the allocation rule obtained according to Fisher's approach can be eas- ily recognized. This means that, for Gaussian populations and equal priors, besides optimizing group separation, Fisher's rule also minimizes the proba- bility of a wrong classification.
In conclusion: in the equal priors case for normal omoschedastic populations, fishers rule is optimal bayes rule.

Important remark 78. Other than hypotizing a MVN distribution of X the f density can be estimated in many different way: a simple way is to use naive bayes.

5.3 Lab

We seen different dataset where we know the groups.

5.3.1 Jobs

A large international air carrier has collected data on employees in three different job classifications:

1. customer service personnel
2. mechanics
3. dispatchers

The director of Human Resources wants to know if these three job classifications appeal to different personality types. Each employee is administered a battery of psychological tests which includes measures of interest in outdoor activity, sociability and conservativeness.

```
jobs <- read.csv("data/jobs.csv", header = TRUE, sep = ";")
head(jobs)

##   outdoor social conservative job
## 1      10     22            5   1
## 2      14     17            6   1
## 3      19     33            7   1
## 4      14     29           12   1
## 5      14     25            7   1
## 6      20     25           12   1

jobs$job <- factor(jobs$job,
                   levels = 1:3,
                   labels = c("Customer service",
                              "Mechanic",
                              "Dispatcher"))
```

The maximum number of useful discriminant functions that can separate the three jobs is less or equal to the minimum between $G - 1$ and p, and so in this case it is less or equal to the minimum between 2 and 3, which is 2.
Thus, here we can find at most 2 useful discriminant functions to separate the employees by their jobs, using the 3 observed variables; these functions are the eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$ which correspond to the not null eigenvalues.

```
## Some info on dimensions
p <- ncol(jobs) - 1
G <- length(unique(jobs$job))
n_tot <- nrow(jobs)   # total number of observation in all the two groups
```

Now we build the sub-datasets: dataset with only employees with each type

```
# we don't want the classification variables here for simplicity
jobs_split <- split(jobs[, -4], jobs$job)
lapply(jobs_split, head, n=2)

## $`Customer service`
##   outdoor social conservative
## 1      10     22            5
## 2      14     17            6
##
## $Mechanic
##    outdoor social conservative
## 86      20     27            6
## 87      21     15           10
##
## $Dispatcher
##     outdoor social conservative
## 179      19     19           16
## 180      17     17           12

n_g <- lapply(jobs_split, nrow)
```

5.3.1.1 Using standard matrix functions

To compute $W^{-1}B$ need to compute within and between group covariance matrix

Withing-group covariance matrix Defined as

$$W = \frac{1}{n-G}\sum_{g=1}^{G} S_g(n_g - 1)$$

we start by calculating the covariance matrix of each group

```
Sg <- lapply(jobs_split, var) # covariance matrix within each group

W <- Reduce(function(x, y) x + y,
            Map(function(S, n) {S * (n - 1)},
                Sg, n_g)) /
  (n_tot - G)
```

Between-group covariance matrix Defined as

$$\mathbf{B} = \sum_{g=1}^{G} n_g (\overline{\mathbf{x}}_g - \overline{\mathbf{x}})(\overline{\mathbf{x}}_g - \overline{\mathbf{x}})^\top$$

```
xbarg <- lapply(jobs_split, colMeans) # average of variable in g-th group
xbar <- colMeans(jobs[, -4]) # overall average

B <- Reduce(function(x, y) x + y,
            Map(function(xb, n) { n * (xb - xbar) %*% t(xb - xbar)},
                xbarg, n_g)) /
  (G - 1)
```

Now, we are able to compute $\mathbf{W}^{-1}\mathbf{B}$ and, more specifically, we can define the (two) discriminant directions through its spectral decomposition.

Discriminant analysis via spectral decomposition Now we do SVD of $W^{-1}B$: we use `eigen`

```
spectral <- eigen(solve(W) %*% B)
round(spectral$values, 2)

## [1] 130.20  38.62   0.00
```

Only the first two are different from zero: this was expected: the maximum number of discriminant functions was 2.
Since we need to separate 3 groups on which 3 variables have been observed, the rank of the matrix $\mathbf{W}^{-1}\mathbf{B}$ is at most equal to 2; in fact there are two not null eigenvalues and two corresponding eigenvectors which are the discriminant coordinates or canonical variables.

Now we extract the canonical variables: the eigenvectors corresponding to not null eigenvalue

```
(A <- spectral$vectors[, 1:2])

##            [,1]       [,2]
## [1,]  0.3470958 -0.9130164
## [2,] -0.7331079 -0.2019912
## [3,]  0.5848738  0.3544018

colnames(jobs[, -4])

## [1] "outdoor"      "social"       "conservative"
```

$\mathbf{A}$ is 3×2 with in rows we have `outdoor`, `social` and `conservative`, while in column the coefficient obtained

Interpretation:

1. look at the coefficiente: the first function has a positive value for first and third, while negative for social. The first discriminant direction is a

linear contrast between interest in social activity and interest in outdoor or conservative activities. This means that the 3 employee categories mainly differ as far as the social component is concerned.

2. the second function has high negative value for outdoor, then a less negative for social and finaly a positive for conservative activities: seems a linear constrast of outtoor vs conservative activity.
   On the second discriminant direction outdoor activities and conservative activities have the largest effect. This direction separates employee categories according to the employees' attitudes towards static activities rather than more dynamic ones.

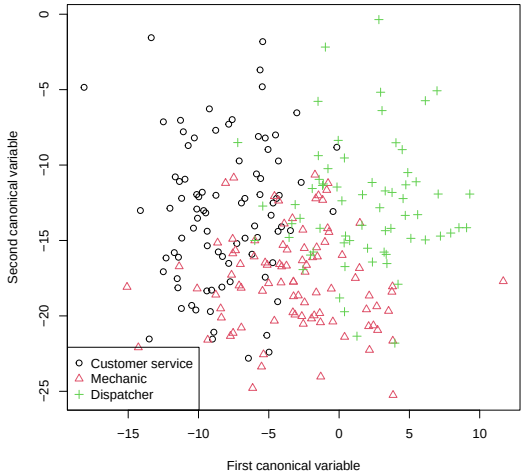
The discriminant directions are unique up to a change of sign. In particular, the sign depends on the normalization constraint imposed by the function used. Namely, eigenvectors obtained through the `eigen` function have unit norm; vectors provided by the `lda` (i.e. a function that will be described in the following) are computed by imposing that the within group covariance matrix is spherical.

Linear combinations Similarly to PC we can find linear comb between our data and the iscriminant function (project original data); we need just to transform our data in matrix

```
X <- as.matrix(jobs[, -4])
Y <- X %*% A # linear combination
```

Thus, each unit is represented by a d-dimensional vector (where $d = rank(W^{-1}B)$). We can plot the obtained linear combinations

```
plot(Y,
     xlab = "First canonical variable",
     ylab = "Second canonical variable",
     col = as.integer(jobs$job),
     pch = as.integer(jobs$job))

legend('bottomleft',
       legend = levels(jobs$job),
       col = 1:3,
       pch = 1:3)
```



These three groups seems not to be very separated according to first two canonical variables

5.3.1.2 Classification based on LDA

Now assume we need to assign a new observation (of unknown group) to one of the three profession.
We compare the distance of each observation to the average of the group, and then assign the observation to the group it is closer to. We can use the euclidean distance on the derived subspace or mahalanobis on the original subspace.

Euclidean distance between new points and the mean vector of the canonical variables First we need to project the average of the group in the derived/discriminant subspaces, that is we need to have $\overline{y}_1, \dots \overline{y}_G$

```
(ybarg <- lapply(xbarg, function(x) x %*% A))

## $`Customer service`
##           [,1]     [,2]
## [1,] -8.136013 -13.1238
##
## $Mechanic
##           [,1]      [,2]
## [1,] -3.132914 -17.60166
##
## $Dispatcher
##          [,1]      [,2]
## [1,] 1.821577 -12.64947
```

The following code returns a matrix containing the Euclidean distances between each projected point from the projected means. We need to calculate the distance from every group, then we will assign to the closest one (this is just for didactical purposes)

```r
## we build a martrix empty at the beginning
euclideanD <- matrix(NA, nrow = n_tot, ncol = G + 1)
# eg in 1,2 the distance between first obs and second group.
for (g in 1:G){
  # the g-th column contains the distance of every row with the g-th group
  euclideanD[, g] <- apply(Y,
                           1,
                           function(y){sqrt(sum((y - ybarg[[g]])^2))})
}

# for each row the observation and in each column the group
# in the last column we put the assigned group (1, 2 or 3)
euclideanD[, G+1] <- apply(euclideanD[, 1:G], 1, which.min)
head(euclideanD) # the last columns has the assigment
```

```
##          [,1]      [,2]      [,3] [,4]
## [1,] 2.073109  8.786252 11.585664    1
## [2,] 4.155567  3.641184  6.088613    2
## [3,] 9.975598 11.090576 17.713411    1
## [4,] 1.774625  7.027652 11.337834    1
## [5,] 2.548430  6.634660 11.517190    1
## [6,] 7.029186  1.908548  8.908554    2
```

Otherwise we can work on the original observations using the Mahalanobis distance.

Mahalanobis distance between each point and the mean vector of each group Working with

$$d_M^2(x_0, \overline{x}_g) = (x_0 - \overline{x}_g)^T W^{-1} (x_0 - \overline{x}_g)$$

Now we do the same with malahanobis distance but on the original variable

```r
## structure is the same of the euclidean, we work on X instead
mahalanobisD <- matrix(NA, nrow = n_tot, ncol = G+1)
for (g in 1:G){
  # the g-th column contains the distance of every row with the g-th group
  # we work with X instead of Y
  mahalanobisD[, g] <- apply(
    X,
    1,
    function(x) t(x - xbarg[[g]]) %*% solve(W) %*% (x - xbarg[[g]])   # check
  )
}
mahalanobisD[, G+1] <- apply(mahalanobisD[, 1:G], 1, which.min)
head(mahalanobisD) # the last columns has the assigment
```

```
##           [,1]     [,2]      [,3] [,4]
## [1,] 1.9551699 6.772098 11.089305    1
## [2,] 3.7381503 3.348347  5.117959    2
```

```
## [3,] 6.8131739 8.986827 21.781321    1
## [4,] 1.9842879 5.148524 10.776851    1
## [5,] 0.5443612 3.178535  9.381197    1
## [6,] 4.0391721 1.139074  6.087088    2
```

Classification rule/missclassification rate In order to check the goodness of our classification and find the misclassification error (i.e. the proportion of observations not correctly assigned) we can compute a cross-table:

```
## elements on the main diagonal indicates units correctly classified
table(jobs$job, mahalanobisD[, 4])

##
##                     1  2  3
##   Customer service 70 11  4
##   Mechanic         16 62 15
##   Dispatcher        3 12 51
```

We can calculate

```
## missclassification rates
mean(as.integer(jobs$job) != mahalanobisD[, 4])

## [1] 0.25

## 25% of obs is wrongly classified
```

5.3.1.3 Using `MASS::lda`

Until here all done manually; to perform the analysis using functions, we can use `MASS::lda` which requires

- `x`: a matrix or data frame or Matrix containing the explanatory variables;
- `grouping`: a factor specifying the true class membership for each observation.

```
(lda_out <- MASS::lda(x = jobs[, -4], grouping = jobs$job))

## Call:
## lda(jobs[, -4], grouping = jobs$job)
##
## Prior probabilities of groups:
## Customer service         Mechanic       Dispatcher
##        0.3483607        0.3811475        0.2704918
##
## Group means:
##                   outdoor   social conservative
```

```
## Customer service 12.51765 24.22353     9.023529
## Mechanic         18.53763 21.13978    10.139785
## Dispatcher       15.57576 15.45455    13.242424
##
## Coefficients of linear discriminants:
##                       LD1         LD2
## outdoor       0.09198065 -0.22501431
## social       -0.19427415 -0.04978105
## conservative  0.15499199  0.08734288
##
## Proportion of trace:
##    LD1    LD2
## 0.7712 0.2288

A # for comparison

##             [,1]       [,2]
## [1,]  0.3470958 -0.9130164
## [2,] -0.7331079 -0.2019912
## [3,]  0.5848738  0.3544018
```

Coefficients of linear discriminants is what we called `A`. Discriminant function should be equivalent up to sign changes: however here results are different from what obtained with single value decomposition of $\mathbf{W}^{-1}\mathbf{B}$: `lda` uses a different constraint put on the within group covariance matrix, set to spherical.
That is usually we have

$$\mathbf{A}^\top \mathbf{W} \mathbf{A} = \mathbf{\Psi}$$

while results from `lda` are given for a A^* such that

$$\mathbf{A}^{*\top} \mathbf{W} \mathbf{A}^* = \mathbf{I}_{G-1}$$
$$\mathbf{A}^* = \mathbf{A}\mathbf{\Psi}^{-1/2}$$

if we want to obtain the same results as lda we need to compute this. To check, for Ψ

```
Psi = t(A) %*% W %*% A  #diagonal matrix (out of are almost zero)
Psi = diag(diag(Psi)) ## to have perfect diag
```

then multiplying A vectors for $\Psi^{-1/2}$ we find again the coefficients of the linear discriminants yield by `lda`

```
(Astar = A %*% solve(Psi^{1/2})) # vector of coefficients

##             [,1]        [,2]
## [1,]  0.09198065 -0.22501431
## [2,] -0.19427415 -0.04978105
## [3,]  0.15499199  0.08734288

lda_out$scaling # the same
```

```
##                      LD1         LD2
## outdoor       0.09198065 -0.22501431
## social       -0.19427415 -0.04978105
## conservative  0.15499199  0.08734288
```

if we use Astar instead of A we get identity

```
t(Astar) %*% W %*% Astar
```

```
##               [,1]          [,2]
## [1,]  1.000000e+00 -6.106227e-16
## [2,] -6.938894e-16  1.000000e+00
```

And the opposite is true, i.e. if we divide the coefficients of the linear discriminants yield by lda by $\Psi^{-1/2}$ we obtain the solution given by the eigen function:

```
coef(lda_out) %*% solve(solve(Psi)^(1/2))
```

```
##                     [,1]       [,2]
## outdoor       0.3470958 -0.9130164
## social       -0.7331079 -0.2019912
## conservative  0.5848738  0.3544018
```

Note: this equivalence holds as far as the conventional definition of matrix B is considered, i.e. groups are weighted by their size in the dataset. The lda function also allows to consider a covariance matrix weighted by the prior probabilities of the classes if these are specified; otherwise observed frequencies are used.

Canonical variates with lda We use `predict` to obtain the projection on the unidimensional space

```
pred <- predict(lda_out) # class, prediction ,posterior is the posterior prob
# matrix and x is the projected units

Ylda <- pred$x
head(Ylda) # should be equal
```

```
##             LD1         LD2
## [1,] -1.6423155  0.71477348
## [2,] -0.1480302  0.15096436
## [3,] -2.6415213 -1.68326115
## [4,] -1.5493681  0.07764901
## [5,] -1.5472314 -0.15994117
## [6,] -0.2203876 -1.07331266
```

```
head(Y) # very different
```

```
##            [,1]      [,2]
```

```
## [1,]  -9.733047 -11.80196
## [2,]  -4.094251 -14.08967
## [3,] -13.503624 -21.53221
## [4,]  -9.382303 -14.38715
## [5,]  -9.374240 -15.35120
## [6,]  -4.367296 -19.05729
```

They are different because use Astar instad of A. also if we try to use Astar does not work because data is centered by lda. So we need to center X to obtain the same results

```
head(scale(X, center = TRUE, scale = FALSE) %*% Astar) # same results
```

```
##            [,1]        [,2]
## [1,] -1.6423155  0.71477348
## [2,] -0.1480302  0.15096436
## [3,] -2.6415213 -1.68326115
## [4,] -1.5493681  0.07764901
## [5,] -1.5472314 -0.15994117
## [6,] -0.2203876 -1.07331266
```

5.3.2 Banknotes

A manager of a bank want to discriminate between genuine and counterfeit banknotes. The following dataset include measures of 100 genuine and 100 false/counterfeit banknotes

```
head(banknotes <- read.csv("data/banknotes.csv", sep = ";"))
```

```
##   Length  Left Right Bottom  Top Diagonal
## 1  214.8 131.0 131.1    9.0  9.7    141.0
## 2  214.6 129.7 129.7    8.1  9.5    141.7
## 3  214.8 129.7 129.7    8.7  9.6    142.2
## 4  214.8 129.7 129.6    7.5 10.4    142.0
## 5  215.0 129.6 129.7   10.4  7.7    141.8
## 6  215.7 130.8 130.5    9.0 10.1    141.4
```

```
## first 100 are true/genuine, last 100 are false
group <- c(rep("Genuine", 100), rep("Counterfeit", 100))
```

We want to separate $G = 2$ groups in which $p = 5$ variables have been observed. The rank of matrix $\mathbf{W}^{-1}\mathbf{B}$ is thus equal to 1. This means that there is only one not null eigenvalue; the corresponding eigenvector is the only discriminant coordinate (or canonical variable)

```
(lda_out <- MASS::lda(x=banknotes, grouping = group)) # only 1 discriminant function
```

```
## Call:
## lda(banknotes, grouping = group)
```

```
##
## Prior probabilities of groups:
## Counterfeit     Genuine
##         0.5         0.5
##
## Group means:
##               Length    Left   Right Bottom    Top Diagonal
## Counterfeit 214.823 130.300 130.193 10.530 11.133  139.450
## Genuine     214.969 129.943 129.720  8.305 10.168  141.517
##
## Coefficients of linear discriminants:
##                   LD1
## Length    0.005011113
## Left      0.832432523
## Right    -0.848993093
## Bottom   -1.117335597
## Top      -1.178884468
## Diagonal  1.556520967
```

We have positive value for left and diagonal and negative for other. The discriminant direction is a linear contrast between the length of the diagonal and the length of the left rim versus height of the right, bottom and top rims. This means that genuine and counterfeit banknotes mainly differ for the length of the diagonal (which summarizes the total dimensions of the notes) and for the length of the left rim with respect to all the other ones.

The linear combination is then $\mathbf{Y} = \mathbf{XA}$ where X is the $n \times p$ data matrix and $\mathbf{A}$ is the $p \times 1$ matrix whose columns are the eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$ corresponding to the non null eigenvalues. Thus, each unit is represented by a one-dimensional vector.

Performing classification Classification can be performed using the predict function. This function classifies multivariate observations by projecting them onto the linear discriminant direction(s) found by lda. It takes as first argument the output of the model fitting and as second one the set of units whose membership has to be predicted. If the latter is equal to the set of units used to fit the model, it can be omitted. The output provides:

- `class`: predicted class membership of each observation;
- `posterior`: posterior probabilities for the classes;
- `x`: scores (observations on the discriminant directions).

```
pred <- predict(lda_out)
```

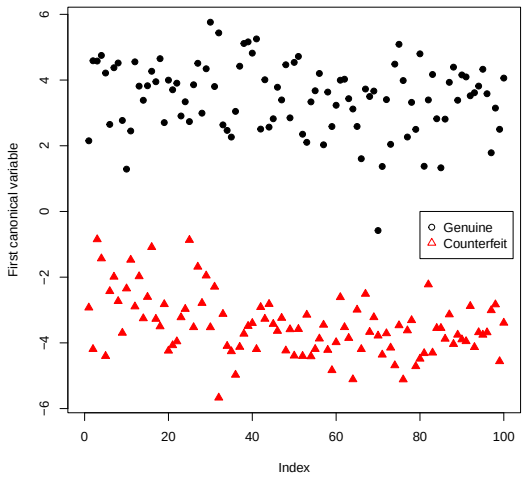
Now we can tabulate the obtained classifications with the true ones:

```
table(group, pred$class) # almost perfect (only 1 banknotes misclassified)
```

```
##
## group         Counterfeit Genuine
##   Counterfeit         100       0
##   Genuine               1      99
```

The classification is almost perfect: only one unit is misclassified probably because it presents an irregular shape close to the original one.
The plot of the units projected onto the space spanned by the canonical variable is obtained as it follows. Banknotes are coloured by the group membership.

```
Y <- pred$x # one dimensional projections
# plot(Y) # not what we want the index separates the groups

# plot all in the 1:100 xlim of index
plot(Y, type = "n", xlim = c(0, 100), # don't plot anything but setup
     ylab = "First canonical variable")
points(Y[1:100], pch = 21, col = "black", bg = "black")
points(Y[101:200], pch = 24, col = "red", bg = "red")
legend(80, 0, c("Genuine", "Counterfeit"),
       col = c("black", "red"), pch = c(21, 24))
```



5.3.3 Wisconsin Diagnostic Breast Cancer (WDBC) Data

This data comes from UCI Machine Learning Repository. It contains 30 regressors, physical measurements of cancer cells, and a binary response containing the diagnosis M (malignant) or B (benign).
We will apply a classifier based on linear discriminant analysis, but we will try to assess its performance by splitting the data in two subsets: a training set and a test set. Indeed, in statistics, goodness of fit refers to how closely a model's predicted values match the observed (true) values. In order to prevent a phenomenon called overfitting (i.e. the production of an analysis that corresponds

too closely or exactly to a particular set of data), it is common to partition the available data in two separate parts. One of them, called training set, is used to estimate the model's parameters, whereas the other, called test set, is employed to validate them.

```
head(wdbc <- read.csv("data/wdbc.csv"))
```

```
##   response    V3    V4     V5     V6      V7      V8     V9     V10    V11
## 1        M 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2419
## 2        M 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812
## 3        M 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2069
## 4        M 11.42 20.38  77.58  386.1 0.14250 0.28390 0.2414 0.10520 0.2597
## 5        M 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1809
## 6        M 12.45 15.70  82.57  477.1 0.12780 0.17000 0.1578 0.08089 0.2087
##       V12    V13    V14   V15    V16      V17     V18     V19     V20     V21
## 1 0.07871 1.0950 0.9053 8.589 153.40 0.006399 0.04904 0.05373 0.01587 0.03003
## 2 0.05667 0.5435 0.7339 3.398  74.08 0.005225 0.01308 0.01860 0.01340 0.01389
## 3 0.05999 0.7456 0.7869 4.585  94.03 0.006150 0.04006 0.03832 0.02058 0.02250
## 4 0.09744 0.4956 1.1560 3.445  27.23 0.009110 0.07458 0.05661 0.01867 0.05963
## 5 0.05883 0.7572 0.7813 5.438  94.44 0.011490 0.02461 0.05688 0.01885 0.01756
## 6 0.07613 0.3345 0.8902 2.217  27.19 0.007510 0.03345 0.03672 0.01137 0.02165
##        V22   V23   V24    V25    V26    V27    V28    V29    V30    V31     V32
## 1 0.006193 25.38 17.33 184.60 2019.0 0.1622 0.6656 0.7119 0.2654 0.4601 0.11890
## 2 0.003532 24.99 23.41 158.80 1956.0 0.1238 0.1866 0.2416 0.1860 0.2750 0.08902
## 3 0.004571 23.57 25.53 152.50 1709.0 0.1444 0.4245 0.4504 0.2430 0.3613 0.08758
## 4 0.009208 14.91 26.50  98.87  567.7 0.2098 0.8663 0.6869 0.2575 0.6638 0.17300
## 5 0.005115 22.54 16.67 152.20 1575.0 0.1374 0.2050 0.4000 0.1625 0.2364 0.07678
## 6 0.005082 15.47 23.75 103.40  741.6 0.1791 0.5249 0.5355 0.1741 0.3985 0.12440
```

```
## divide the dataset using sample
set.seed(19)
n <- nrow(wdbc)
train <- sample(n, size = n * 2/3, replace =FALSE) # 2/3 to train  1/3 to test

wdbc_train <- wdbc[train, ]
wdbc_test <- wdbc[-train, ]
```

Partitioning the dataset

```
lda_out <- MASS::lda(x = wdbc_train[, -1], grouping = wdbc_train[, 1])
```

Compute discriminant directions by lda Predictions for the training data and wrong missclassification in the training set

```
class_train <- predict(lda_out)$class
C <- table(class_train, wdbc_train[, 1])
1 - sum(diag(C)) / sum(C)

## [1] 0.02902375
```

Predictions for the test data and right missclassification in the test set (we see error is higher, but right, compared to previous one)

```
class_test <- predict(lda_out, newdata = wdbc_test[, -1])$class
C <- table(class_test, wdbc_test[, 1])
1 - sum(diag(C)) / sum(C)

## [1] 0.07894737
```