

Data Analysis

Christophe Ambroise



Section 1

Introduction

Data Analysis is a family of methods (sometimes statistical) allowing to explore data represented mostly in the form of

- array where each line describes an object (individual) and each column a variable:
- of similarity (or dissimilarity) pair between objects

Certain methods, mainly geometric, make it possible to highlight the relations which can exist between the various data and to establish a summary (sometimes with statistic characteristic).

In French, the terminology “analyse de données” refers to a subset of what is more commonly called multivariate statistics.

The purpose of the data analysis is exploratory and aims to

- visualize or
- summarize.

Both aspects are related. This course presents some of the most common methods.

Section 2

A few introductory examples

- Data: $(X_i)_i$ the gray levels of an image
- Assumption there are $(Z)_i$ hidden label of the K gray levels classes

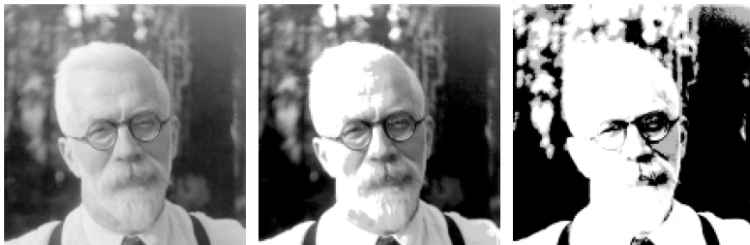


Figure 1: Famous Statistician R. Fisher

Data

- V a set of given nodes $\in \{1, \dots, n\}$,
- E a set of edges $\in \{1, \dots, n\}^2$,
- $\mathbf{X} = (X_{ij})$ the adjacency matrix such that $\{X_{ij} = 1\} = \mathbb{I}\{i \leftrightarrow j\}$.
 - oriented : $X_{ij} \neq X_{ji}$,
 - valued : $X_{ij} \in \mathbb{R}$.

Assumption

hyp.: there exists a hidden structure into Q classes of connectivity,

- $\mathbf{Z} = (\mathbf{Z}_i)_i$, $Z_{iq} = \mathbb{I}\{i \in q\}$ are indep. hidden variables,
- $\alpha = \{\alpha_q\}$, the *prior* proportions of groups,

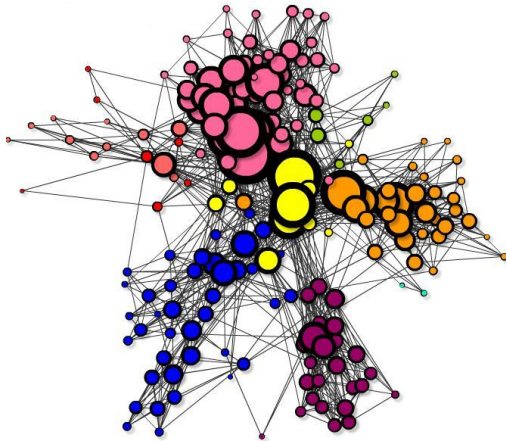


Figure 2: Sample of 250 blogs (nodes) with their links (edges) of the French political Blogosphere

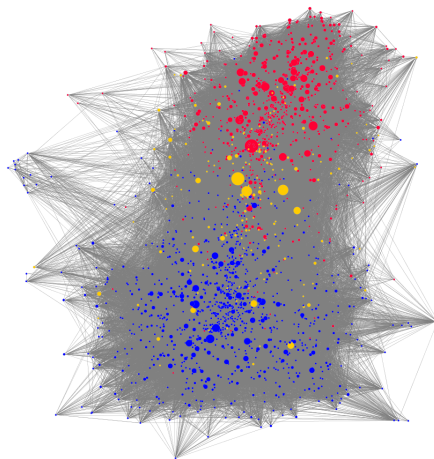


Figure 3: US political blog

In the Nature article "Gene Mirror Geography Within Europe", Novembre et al., 2008,

Data

- 3000 individuals
- described by 500,000 SNP (Single Nucleotide Polymorphism)

can be represented by a data table $\mathbf{X} = (X_{ij})$ with

- 3000 rows (if 1 meter high)
- 50,000 columns (then 166 meters large !!!)

- 90 % of human genetic variation,
- SNP with allelic frequency greater than 1 % are present every 300 base pairs in average (in human genome)

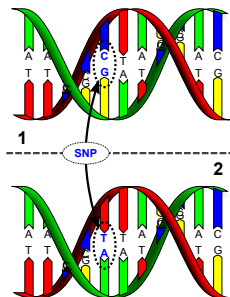


Figure 4: SNP (wikipedia)

Mother	Father	
	C	T
C	0	1
T	1	2

- How to summarize this table
- Each person is a vector in a 500,000 dimensional space

Solution: Principal component Analysis

Summarize 500,000 columns by 2 columns (linear combination of the original)

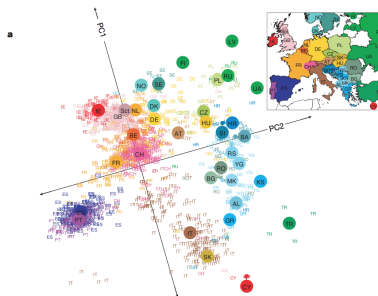


Figure 5: PCA of european population

Section 3

Outline and evaluation of the course

- ➊ Multivariate gaussian
- ➋ Clustering (Discrete latent variable)
 - partition and kmeans
 - hierarchical clustering
 - Mixture model (EM), Gaussian and other simple mixture models (Bernoulli)
- ➌ Continuous Latent Variable :
 - Principal Component Analysis (PCA)
 - Probabilistic PCA
 - Kernel PCA

- 1 R project (1/3)
- 2 Final Exam (2/3)

Section 4

Probability reminder

Random Variables and their realization are both denoted with lower cases. The context should always allow to distinguish between both.

- Expectation: the average value of some function $f(x)$ under a probability distribution $p(x)$;
- discrete case: $E[f] = \sum_x p(x)f(x)$
- continuous case: $E[f] = \int p(x)f(x)dx$
- using N points drawn from the prob. distribution or prob. density, expectation can be approximated by:

$$E[f] \approx \frac{1}{N} \sum_{i=1}^N f(x_i)$$

- conditional expectation (discrete case): $E[f|y] = \sum_x p(x|y)f(x)$
- Variance of $f(x)$: a measure of the variations of $f(x)$ around $E[f]$.
- $var[f] = E[(f(x) - E[f(x)])^2]$

Expectations and covariances II

Exercise

Show that

$$\text{var}[f] = E[(f(x) - E[f(x)])^2] = E[f^2] - E[f]^2$$

- $\text{var}[f] = E[f^2] - E[f]^2$
- $\text{var}[x] = E[x^2] - E[x]^2$
- Covariance for two random variables:

$$\text{cov}[x, y] = E_{x,y}[xy] - E[x]E[y]$$

Exercise

Show that if two variable are independent, then there covariance is zero

- Variance of the sum of two random variables

$$\text{var}[x + y] = \text{var}[x] + \text{var}[y] + 2\text{cov}(x, y)$$

Expectations and covariances III

- Expectation of a random vector

$$E[\mathbf{x}] = (E[x_i])_i$$

- Two vectors of random variables:

$$\text{cov}[\mathbf{x}, \mathbf{y}] = E_{\mathbf{x}, \mathbf{y}}[\mathbf{x}\mathbf{y}^t] - E[\mathbf{x}]E[\mathbf{y}^t]$$

- Covariance matrix of a random vector

$$\text{var}[\mathbf{x}] = (\text{cov}(x_i, x_j))_{ij}$$

Exercise

Let A be an $p \times p$ matrix and \mathbf{x} a random vector. Show that

$$\text{var}[A\mathbf{x}] = A\text{var}[\mathbf{x}]A^t.$$

Section 5

Multivariate normal distribution

Univariate normal distribution I

The random variable x is Gaussian with mean μ and variance σ^2 when it has a probability density

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2}.$$

We note $\mathcal{N}(\mu, \sigma^2)$.

Graphically, the average represents the position of the curve on the abscisse axis and the standard deviation its crushing (the area under the curve being obviously the unit).

```
library(ggplot2)
f <- ggplot(data.frame(x = c(-4, 4)), aes(x))
f + stat_function(fun = dnorm, colour = "cornflowerblue") +
  stat_function(fun = dnorm, colour = "coral", args=list(sd=2))
```

Univariate normal distribution II

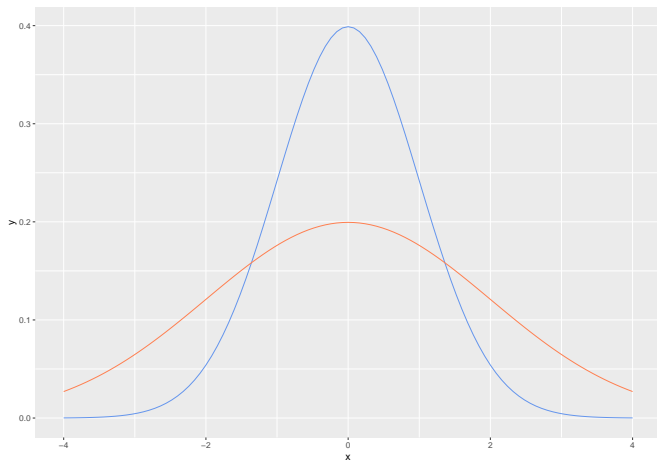


Figure 6: Two gaussian univariate distribution with standart deviation 1 (blue) and 2 (orange)

Univariate normal distribution III

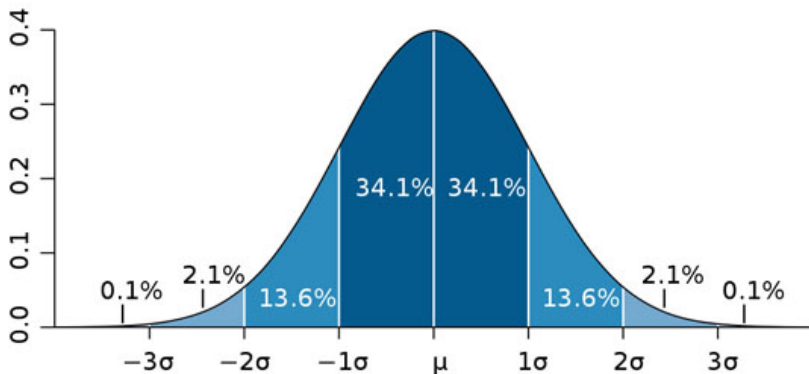


Figure 7: Gaussian distribution mean and standard deviation

Probability of a gaussian random variable to be in a interval

Compute the probability of $X \sim \mathcal{N}(\mu, \sigma^2)$ to be in $[\mu - k\sigma, \mu + k\sigma]$ for $k \in \{1, 2, 3, 4\}$

If we have a sample (x_1, \dots, x_n) i.i.d. realization of a variable random normal $\mathcal{N}(\mu, \sigma^2)$, the classical (frequentist) strategy of estimating parameters consists in choosing the values of the parameters that maximize the likelihood function:

$$p((x_1, \dots, x_n)|\mu, \sigma^2) = \prod_i \mathcal{N}(x_i|\mu, \sigma^2).$$

The log-likelihood can be written in the form

$$\ln p((x_1, \dots, x_n)|\mu, \sigma^2) = -\frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2 - \frac{n}{2} \ln \sigma^2 - \frac{n}{2} \ln 2\pi.$$

Exercise

Show that maximum likelihood estimators are therefore the empirical mean

$$\hat{\mu}_{ml} = \frac{1}{n} \sum_i x_i$$

and the empirical variance

$$\hat{\sigma}_{ml}^2 = \frac{1}{n} \sum_i (x_i - \hat{\mu}_{ml})^2 = \frac{1}{n} \sum_i x_i^2 - \hat{\mu}_{ml}^2.$$

It is immediate to see that the two estimators are convergent but that the variance estimator is biased downwards:

$$\mathbb{E}[\hat{\mu}_{ml}] = \mu, \quad \mathbb{E}[\hat{\sigma}_{ml}^2] = \frac{n-1}{n} \sigma^2.$$

Multivariate normal distribution I

The density of a random normal vector $\mathbf{x} = (x^1, \dots, x^p)'$ with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is defined as

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right\}$$

where $\mathbf{x} = (x^1, \dots, x^p)'$.

Properties

- 1 Any sub-vector of this random vector still follows a normal distribution; in particular, the variables x^1, \dots, x^p are all gaussian.
- 2 For a Gaussian random vector, the variables x^1, \dots, x^p are independent iff the covariance matrix is diagonal.
- 3 if $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then $A\mathbf{x} + \mathbf{b} \sim \mathcal{N}(A\boldsymbol{\mu} + \mathbf{b}, A\boldsymbol{\Sigma}A')$

Characterization of the distribution I

Density is a function of \mathbb{R}^p in \mathbb{R} ; it is possible to characterize such a density using equidensity sets $f(\mathbf{x}) = cste$. For the normal law, these sets are ellipsoids from \mathbb{R}^p . The mean vector μ and the covariance matrix then correspond to the center and shape of these ellipsoids.

It is possible to have a geometric intuition based on the spectral decomposition

$$\Sigma = \lambda D A D'$$

where λ is a positive real, D is an orthogonal matrix and A a diagonal matrix of determinant 1. This decomposition can be interpreted as follows: λ characterizes the volume, D the direction and A the form of the distribution. We can thus define a matrix of variance from these three characteristics.

For example, in the case of \mathbb{R} , D is a rotation matrix defined by an angle Θ and A is a diagonal matrix of diagonal terms a and $1/a$.

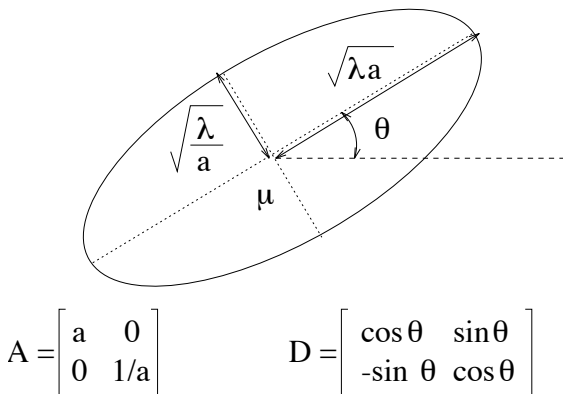


Figure 8: Paramétrisation de la matrice de variance dans le plan

Exercice

Show that :

If the random variables x^1, \dots, x^p are independent, normal of mean 0 and variance 1, then the random vector $\mathbf{x} = (x^1, \dots, x^p)'$ is normal, average 0, and variance I_p . Indeed

$$\begin{aligned}f(\mathbf{x}) &= f(x^1) \dots f(x^p) && \text{(indépendance)} \\&= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x^1)^2\right) \dots \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x^p)^2\right) \\&= \frac{1}{(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}((x^1)^2 + \dots (x^p)^2)\right) \\&= \frac{1}{(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}\mathbf{x}'I\mathbf{x}\right).\end{aligned}$$

if matrix Σ can be decomposed as $\Sigma = TT'$ where T is a $(p \times p)$ matrix, then the transformation $y = Tx + \mu$ transforms X following $\mathcal{N}(0, I)$ into a random vector following $\mathcal{N}(\mu, \Sigma)$. It is enough to apply the property (3): one obtains a normal distribution with mean vector $T \cdot 0 + \mu = \mu$ and covariance matrix $TIT' = TT' = \Sigma$.

Determination of T

First, note that if there is a solution, it is not unique: indeed, if T checks $TT' = \Sigma$ and if U is an orthogonal matrix, then $S = TU$ checks the same property.

The Cholesky decomposition $\Sigma = TT'$ where T is a triangular matrix is a solution; we can for example use the function `chol` in R. We can also easily show that the matrix

$$T = \sqrt{\lambda} D \sqrt{A} D'$$

defined from the previous decomposition $\Sigma = \lambda D A D'$ is another solution.

Maximum likelihood estimators I

In the multivariate case the log-likelihood of the parameters μ, Σ in view of a sample of n vectors of dimension p ($\mathbf{x}_1, \dots, \mathbf{x}_n$) expresses itself as

$$\ln p(\mathbf{x}_1, \dots, \mathbf{x}_n | \mu, \Sigma) = -\frac{1}{2} \left(n \ln(|\Sigma|) + \sum_i (\mathbf{x}_i - \mu)^T \Sigma^{-1} (\mathbf{x}_i - \mu) + np \ln(2\pi) \right) \quad (1)$$

$$= -\frac{n}{2} (\ln(|\Sigma|) + \text{tr}(\Sigma^{-1} \hat{\Sigma}) + p \ln(2\pi)) \quad (2)$$

with

$$\hat{\Sigma} = \frac{1}{n} \sum_i (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T.$$

Maximum likelihood estimators are thus

$$\hat{\mu}_{ml} = \frac{1}{n} \sum_i \mathbf{x}_i,$$

and

$$\hat{\Sigma}_{ml} = \frac{1}{n} \sum_i (\mathbf{x}_i - \hat{\mu}_{ml})(\mathbf{x}_i - \hat{\mu}_{ml})^T.$$

Elements of proof

$$\frac{\partial(b^T a)}{\partial a} = b$$

$$\frac{\partial(a^T Aa)}{\partial a} = (A + A^T)a$$

$$\frac{\partial}{\partial A} \text{tr}(BA) = B^T$$

$$\frac{\partial}{\partial A} \log |A| = (A^{-1})^T$$

$$\text{tr}(ABC) = \text{tr}(CAB) = \text{tr}(BCA)$$

Thus if x is a vector

$$x^T Ax = \text{tr}(x^T Ax) = \text{tr}(Axx^T)$$

Derivation

$$\frac{\partial}{\partial \boldsymbol{\mu}} (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) = \frac{\partial \mathbf{y}_i}{\partial \boldsymbol{\mu}} \frac{\partial \mathbf{y}_i^T \boldsymbol{\Sigma}^{-1} \mathbf{y}_i}{\partial \mathbf{y}_i} \\ - 1(2\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})),$$

Hence

$$\frac{\partial}{\partial \boldsymbol{\mu}} \ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{\Sigma}^{-1} \sum_i (\mathbf{x}_i - \boldsymbol{\mu}) = 0$$

and

$$\hat{\boldsymbol{\mu}}_{ml} = \frac{1}{n} \sum_i \mathbf{x}_i,$$

We have

$$\frac{\partial \ell(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \frac{n}{2} \log |\mathbf{\Lambda}| - \frac{n}{2} \text{tr}(\hat{\mathbf{\Sigma}} \mathbf{\Lambda}) = 0$$

Hence

$$\mathbf{\Lambda} = \hat{\mathbf{\Sigma}}^{-1}$$

and eventually

$$\hat{\mathbf{\Sigma}}_{ml} = \frac{1}{n} \sum_i (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{ml})(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_{ml})^T.$$

Let x_1, \dots, x_k k reduced centered Gaussian variables:

$$Q = \sum_{i=1}^k x_i^2,$$

is distributed according to a Chi 2 law at k degree of freedom. We notice:

$$Q \sim \chi^2(k) \text{ or } Q \sim \chi_k^2.$$

if $\mathbf{x} \sim \mathcal{N}_p(\boldsymbol{\mu}, \Sigma)$ then $\mathbf{y} = \Sigma^{-1/2}(\mathbf{x} - \boldsymbol{\mu}) \sim \mathcal{N}_p(0, I_p)$ and

$$Q = \mathbf{y}^t \mathbf{y} \sim \chi_p^2$$

The equation

$$P(Q \leq q) = \alpha$$

where $q = \chi_{p,\alpha}^2$ defines a α level equiprobability ellipsoid.

Cochran Theorem (simplified version) I

Let \mathbf{x} be a Gaussian vector of p -mean vector zero and matrix of variance identity:
 $\mathbf{x} \sim \mathcal{N}_p(0, I_p)$.

Let E be a vector space of dimension $r < p$ and P_E the orthogonal projection on E and P_{E^\perp} the projection on the supplementary orthogonal of E .

The projections $P_E(\mathbf{x})$ and $P_{E^\perp}(\mathbf{x})$ are independent Gaussian law vectors

$$P_E \mathbf{x} \sim \mathcal{N}_p(0, P_E) \text{ et } P_{E^\perp} \mathbf{x} \sim \mathcal{N}_p(0, P_{E^\perp}).$$

Projection norms are random variables that follow laws of the χ^2 :

$$\|P_E \mathbf{x}\|^2 \sim \chi_r^2 \text{ et } \|P_{E^\perp} \mathbf{x}\|^2 \sim \chi_{p-r}^2.$$

Demonstration $\{-\}$ I

Let $B = [\mathbf{b}_1 \cdots \mathbf{b}_p]$ an orthonormal basis of \mathbb{R}^p such that $[\mathbf{b}_1 \cdots \mathbf{b}_r]$ be an orthonormal basis of E and $[\mathbf{b}_{r+1} \cdots \mathbf{b}_p]$ a basis of E^\perp .

Note P the matrix of passage from the canonical base to the base B .

Let $\mathbf{y} = y_1, \dots, y_r$ the coordinates of \mathbf{x} in the orthonormal basis $[\mathbf{b}_1 \cdots \mathbf{b}_r]$ from E :

$$P_E(\mathbf{x}) = y_1 \mathbf{b}_1 + \dots + y_r \mathbf{b}_r = J_r \mathbf{y}, \quad (3)$$

$$P_{E^\perp} \mathbf{x} = y_{r+1} \mathbf{b}_{r+1} + \dots + y_p \mathbf{b}_p = J_{p-r} \mathbf{y}, \quad (4)$$

where J_r is a diagonal matrix with 1 on the r first diagonal coefficients and 0 afterwards, and $J_{p-r} = I_p - J_r$.

We have by definition $\mathbf{y} = P^t \mathbf{x}$. And according to the remark of the subsection Gaussian vector and orthonormal basis: $\mathbf{y} \sim \mathcal{N}_p(0, I_p)$. The two projections are therefore independent because combinations linear of independent Gaussian variables.

We can then go back to the vector \mathbf{x} by noticing that $P_E \mathbf{x} = P J_r \mathbf{y}$ and $P_{E^\perp} \mathbf{x} = P J_{p-r} \mathbf{y}$ two Gaussian centered vector of respective covariance matrix P_E and P_{E^\perp} .

And if we consider the standard of projection:

$$\|P_E(\mathbf{x})\|^2 = \sum_{j=1}^r y_j^2 \sim \chi_r^2 \quad (5)$$

$$\|P_{E^\perp}(\mathbf{x})\|^2 = \sum_{j=r+1}^p y_j^2 \sim \chi_{(p-r)}^2 \quad (6)$$

Application to mean and empirical variance I

Let $\mathbf{x} = (x_1, \dots, x_n)$, n be independent gaussian variables $\mathcal{N}(0, \sigma^2)$.

We have $\mathbf{x} - \mu \mathbb{I}_n \sim \mathcal{N}(0, \sigma^2 I_n)$.

Let E the vector space generated by the vector $\frac{1}{\sqrt{n}} \mathbb{I}_n = \frac{1}{\sqrt{n}}(1, \dots, 1)^t$ and E^\perp its complementary in \mathbb{R}^n .

The projection of $\mathbf{x} - \mu \mathbb{I}_n$ on E is the vector

$$P_E(\mathbf{x} - \mu \mathbb{I}_n) = \frac{1}{\sqrt{n}} \mathbb{I}_n^t (\mathbf{x} - \mu) \frac{1}{\sqrt{n}} \mathbb{I}_n = (\bar{x} - \mu \mathbb{I}_n) \mathbb{I}_n$$

The projection of $\mathbf{x} - \mu \mathbb{I}_n$ on E^\perp is the vector:

$$P_{E^\perp}(\mathbf{x} - \mu \mathbb{I}_n) = (\mathbf{x} - \mu \mathbb{I}_n) - P_E(\mathbf{x} - \mu \mathbb{I}_n) = \begin{pmatrix} x_1 - \bar{x} \\ x_2 - \bar{x} \\ \vdots \\ x_n - \bar{x} \end{pmatrix}$$

Both projections are independent (orthogonal). Standard projections are variables that follow $\sigma^2\chi^2$ of respective degree 1 and $n - 1$:

$$n(\bar{x} - \mu)^2 \sim \sigma^2\chi_1^2,$$

$$\sum (x_i - \bar{x})^2 \sim \sigma^2\chi_{n-1}^2.$$

R has several functions to simulate variables samples random. For example, the `rbinom`, `rnorm`, `rpois`, `runif` commands allow to simulate samples of binomial, normal, random variables of Fish, and uniforms.

The multivariate normal law can be directly simulated with the `mvrnorm` {MASS} function or simulated from an independent Gaussian variable with the `rnorm` {stats} function (see the exercises).

Section 6

Le langage R de base (R base)

Qu'est-ce que R ?

En bref

R est un logiciel de développement scientifique spécialisé dans le calcul et l'**analyse statistique**

Mais aussi

- un langage interprété,
- un environnement de développement,
- un projet open source (projet **GNU**),
- un logiciel multi-plateforme (Linux, Mac, Windows),
- la 18ième lettre de l'alphabet

- ➊ Gestionnaire de données - Lecture, manipulation, stockage.
- ➋ Algèbre linéaire - Opérations classiques sur vecteurs, tableaux et matrices
- ➌ Statistiques et analyse de données - Dispose d'un *grand* nombre de méthodes d'analyse de données (des plus anciennes et aux plus récentes)
- ➍ Moteur de sorties graphiques - Sorties écran ou fichier
- ➎ Système de modules - Alimenté par la communauté (+ de 2000 extensions!)
- ➏ Interface facile avec C/C++, Fortran

Chronologie

- 1970s** développement de S au Bell labs.
- 1980s** développement de S^+ au AT&T. Lab
- 1993** développement de R sur le modèle de par Robert Gentleman et Ross Ihaka au département de statistique de l'université d'Auckland.
- 1995** dépôts des codes sources sous licence GNU/GPL
- 1997** élargissement du groupe
- 2002** la fondation dépose ses statuts sous la présidence de Gentleman et Ihaka
- 2010** Les débuts de Rstudio

❶ La page web de la **fondation**

- les statuts, des liens, des références.
- `<http://www.r-project.org/>`

❷ La page web du **CRAN** (Comprehensive R Arxiv Network)

- binaires d'installation, packages, documentations, ...
- `<http://cran.r-project.org/>`

❸ La **conférence** des utilisateurs de R:

- annuelle, prochaine édition à Toulouse
- `http://www.user2019.fr/`

❹ **The R journal** propose des articles sur

- de nouvelles extensions, des applications, des actualités.
- `http://journal.r-project.org/`

Plus

- Libre et gratuit,
- Richesse des modules (en statistique),
- Rapidité d'exécution,
- Développement rapide (langage de scripts),
- Syntaxe intuitive et compact,
- Nombreuses possibilités graphiques.

Moins

- Aide intégrée succincte,
- Debugger un peu sec,
- Code parfois illisible (compacité),
- Personnalisation des graphiques un peu lourde.

Les logiciels de développement scientifique sont spécialisés en

❶ algèbre linéaire

- Matlab de Mathworks, une référence,
- Scilab de l'INRIA, l'alternative libre de matlab,
- Octave de GNU, l'alternative open source ,

❷ statistiques

- SAS (SAS Inc.), la référence,
- S-PLUS (TIBCO), le concurrent,

❸ calcul symbolique

- Mathematica (Wolfram), la référence,
- Mapple (Maplesoft), la référence aussi,
- Maxima (GNU), l'alternative open source .

❹ Autres

- Julia
- Python, le concurrent le plus sérieux ?,

- Anciens ouvrages de référence
- R base, un livre de base
- Nouvelle mode: Wickam et co

Section 7

R Studio comme interface

- Il fournit un éditeur intégré,
- fonctionne sur toutes les plates-formes (y compris sur des serveurs) et
- offre de nombreux avantages tels que l'intégration avec la version contrôle et gestion de projet.

Lorsque vous ouvrez RStudio pour la première fois, vous serez accueilli par trois panneaux:

- La console interactive R (entier gauche)
- Environnement / Histoire (onglet en haut à droite)
- Files / Plots / Packages / Help / Viewer (onglet en bas à droite)

Une fois que vous ouvrez des fichiers, tels que des scripts R, un panneau d'éditeur s'ouvre également en haut à gauche.

Il existe deux manières principales d'interagir avec R:

- en utilisant la console
- ou en utilisant des fichiers de script (fichiers texte contenant votre code).

La fenêtre de la console (dans RStudio, le panneau en bas à gauche)

- endroit où R vous attend pour lui dire quoi faire et où il montrera les résultats d'une commande
- Vous pouvez taper des commandes directement dans la console,
- mais elles seront oubliées lorsque vous fermerez la session.

- Il est préférable d'entrer les commandes dans l'éditeur de script
- et de sauvegarder le script
- envoyer la ligne actuelle ou le texte sélectionné à la console R à l'aide du raccourci Ctrl-Entrée

Raccourcis console - éditeur

Ctrl-1 et Ctrl-2 qui vous permettent de sauter entre le script et les fenêtres de la console.

- Utilisez # signes pour commenter.
- Commentez libéralement dans vos scripts R.
- Tout ce qui se trouve à droite d'un # est ignoré par R.

- RStudio vous offre une grande souplesse dans l'exécution du code depuis l'éditeur fenêtre. Il existe des boutons, des choix de menus et des raccourcis clavier. Pour exécuter la ligne en cours, vous pouvez
 - ❶ cliquez sur le bouton `Run` situé au-dessus du panneau de l'éditeur,
 - ❷ sélectionnez "Run Lines" dans le menu "Code", ou
 - ❸ appuyez sur `Ctrl-Entrée` dans Windows ou Linux ou sur `Commande-Entrée` sur OS X. (Ce raccourci peut également être vu en survolant la souris sur le bouton).

- Pour lancer un bloc de code, sélectionnez-le, puis Exécuter.
- Si vous avez modifié une ligne de code dans un bloc de code que vous venez d'exécuter, il n'est pas nécessaire de resélectionner la section et Run, vous pouvez utiliser le bouton suivant,
- Ré-exécuter la région précédente Cela exécutera le bloc de code précédent inculquer les modifications que vous avez apportées.

La chose la plus simple que vous puissiez faire avec R est l'arithmétique:

```
1 + 100
```

```
## [1] 101
```

Et R imprimera la réponse, avec un précédent “[1]”. Ne t’inquiète pas pour ça pour l’instant, nous l’expliquerons plus tard. Pour l’instant, considérez-le comme indiquant une sortie.

Comme bash, si vous tapez une commande incomplète, R vous attendra pour complète le:

```
> 1 +
```

```
+
```

Chaque fois que vous appuyez sur Entrée et que la session R affiche un “+” au lieu d’un “>”, signifie qu’il attend que vous complétiez la commande. Si vous voulez annuler une commande que vous pouvez simplement appuyer sur “Esc” et RStudio vous rendra le “>” rapide.

- Si vous utilisez R depuis la ligne de commande plutôt que depuis RStudio,
- vous devez utiliser `Ctrl + C` au lieu de `Esc` pour annuler la commande. Ce s'applique également aux utilisateurs de Mac!
- Annuler une commande n'est pas seulement utile pour tuer des commandes incomplètes:
- vous pouvez aussi l'utiliser pour dire à R d'arrêter d'exécuter du code (par exemple si
- en prenant beaucoup plus de temps que prévu, ou pour se débarrasser du code que vous êtes en train d'écrire.

Ordre des opérations I

Lorsque vous utilisez R comme calculatrice, l'ordre des opérations est le même que vous auriez appris à l'école.

De la plus haute à la plus basse préséance:

- Parenthèses: (,)
- Exposants: ^ ou **
- Diviser: /
- Multiplier: *
- Ajouter: +
- Soustraire: -

```
3 + 5 * 2
```

```
## [1] 13
```

Utilisez des parenthèses pour regrouper les opérations afin de forcer l'ordre de évaluation si elle diffère du défaut, ou pour préciser ce que vous avoir l'intention

```
(3 + 5) * 2
```

```
## [1] 16
```

Ordre des opérations II

Cela peut devenir compliqué lorsque cela n'est pas nécessaire, mais clarifie vos intentions. Rappelez-vous que d'autres peuvent lire votre code ultérieurement.

```
(3 + (5 * (2 ^ 2))) # difficile à lire  
3 + 5 * 2 ^ 2 # si vous vous souvenez des règles  
3 + 5 * (2 ^ 2) # Si vous oubliez certaines règles, cela pourrait vous aider
```

Le texte après chaque ligne de code s'appelle un "commentaire". Tout ce qui suit le symbole de hachage (ou octothorpe) "#" est ignoré par R lorsqu'il exécute du code.

Les nombres vraiment petits ou grands ont une notation scientifique:

```
2/10000
```

```
## [1] 2e-04
```

Ce qui est un raccourci pour "multiplié par 10^{-4} ". Donc $2e-4$ est un raccourci pour $2 * 10^{-4}$.

Vous pouvez aussi écrire des nombres en notation scientifique:

```
5e3 # Notez l'absence de moins ici
```

```
## [1] 5000
```

R a beaucoup de fonctions mathématiques intégrées. Pour appeler une fonction, nous tapons simplement son nom, suivi par des parenthèses ouvertes et fermantes. Tout ce que nous tapons à l'intérieur des parenthèses s'appelle la fonction arguments:

```
sin (1) # fonctions trigonométriques
```

```
## [1] 0.841471
```

```
log (1) # logarithme naturel
```

```
## [1] 0
```

```
log10 (10) # base-10 logarithme
```

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

```
exp (0.5) #  $e^{(1/2)}$ 
```

```
## [1] 1.648721
```

Ne vous souciez pas d'essayer de vous souvenir de toutes les fonctions de R:

- rechercher sur Google,
- ou si vous vous souvenez de la début du nom de la fonction, utilisez *complétion* dans RStudio.

C'est un avantage que RStudio sur R , il dispose de *capacités d'auto-complétion* qui vous permettent plus facilement rechercher des fonctions, leurs arguments et les valeurs qu'ils prendre.

Taper un ? Avant le nom d'une commande ouvrira la page d'aide pour cette commande.
En plus de fournir

- une description la page d'aide affichera généralement
- une collection d'exemples

Comparer les choses

Nous pouvons également faire la comparaison en R:

```
1 == 1 # égalité (noter deux signes égaux, lire comme "est égal à")
```

```
## [1] TRUE
```

```
1 != 2 # inégalité (lire comme "n'est pas égal à")
```

```
## [1] TRUE
```

```
1 < 2 # moins que
```

```
## [1] TRUE
```

```
1 <= 1 # inférieur ou égal à
```

```
## [1] TRUE
```

```
1 > 0 # plus grand que
```

```
## [1] TRUE
```

```
1 >= -9 # supérieur ou égal à
```

```
## [1] TRUE
```


- Un mot d'avertissement sur la comparaison des chiffres: vous devriez ne jamais utiliser `==` pour comparer deux nombres à moins qu'ils ne soient entiers (un type de données pouvant représenter spécifiquement uniquement des nombres entiers).
- Les ordinateurs ne peuvent représenter que des nombres décimaux avec un certain degré de précision, donc deux nombres qui semblent le même lorsqu'il est imprimé par R, peut effectivement avoir différentes représentations sous-jacentes et donc être différent par une petite marge d'erreur (appelée Machine tolérance numérique).
- Au lieu de cela, vous devez utiliser la fonction `all.equal`.
- Lectures complémentaires: <http://floating-point-gui.de/>

Stocker des valeurs dans des variables en utilisant l'opérateur d'affectation `<-`

```
x <- 1/40
```

```
x
```

```
## [1] 0.025
```

Plus précisément, la valeur stockée est une approximation * décimale * de cette fraction appelée [nombre à virgule flottante] (http://en.wikipedia.org/wiki/Floating_point).

Recherchez l'onglet `Environment` dans l'un des volets de RStudio, et vous verrez que `x` et sa valeur est apparu. Notre variable `x` peut être utilisée à la place d'un nombre dans tout calcul qui attend un nombre:

```
log (x)
```

```
## [1] -3.688879
```

Notez également que les variables peuvent être réaffectées:

```
x <- 100
```

x contenait la valeur 0.025 et a maintenant la valeur 100.

Les valeurs d'affectation peuvent contenir la variable affectée à:

```
x <- x + 1 #notice comment RStudio met à jour sa description de x en haut
```

Le côté droit de l'affectation peut être toute expression R valide. Le côté droit est *entièrement évalué* avant que l'affectation ait lieu.

Les noms de variables peuvent contenir des lettres, des chiffres, des traits de soulignement et des points. Ils ne peuvent pas commencer avec un nombre ni contenir des espaces du tout. Différentes personnes utilisent différentes conventions pour les noms de variables longues, celles-ci comprennent

- `periods.between.words`
- `souligne_entre_mots`
- `camelCaseToSeparateWords`

Ce que vous utilisez dépend de vous, mais **soyez cohérent**.

Il est également possible d'utiliser l'opérateur = pour l'affectation:

```
x = 1/40
```

Mais c'est beaucoup moins courant parmi les utilisateurs de R.

Donc, la recommandation est d'utiliser <=.

Il existe quelques commandes utiles que vous pouvez utiliser pour interagir avec la session R.

`ls` listera toutes les variables et fonctions stockées dans l'environnement global (votre session de travail):

```
ls()
```

```
## [1] "f" "x"
```

- Comme dans le shell, `ls` cachera toutes les variables ou fonctions commençant avec un `"."` par défaut.
- Pour lister tous les objets, tapez `ls (all.names = TRUE)`

Remarque

Notez ici que nous n'avons donné aucun argument à `ls`, mais nous avons quand même nécessaire de donner aux parenthèses de dire à R d'appeler la fonction.

Contenu d'une fonction

Si vous tapez `ls` par lui-même, R imprimera le code source de cette fonction!

```
ls
```

```
## function (name, pos = -1L, envir = as.environment(pos), all.names = FALSE,
##   pattern, sorted = TRUE)
## {
##   if (!missing(name)) {
##     pos <- tryCatch(name, error = function(e) e)
##     if (inherits(pos, "error")) {
##       name <- substitute(name)
##       if (!is.character(name))
##         name <- deparse(name)
##       warning(gettextf("%s converted to character string",
##         sQuote(name)), domain = NA)
##       pos <- name
##     }
##   }
##   all.names <- .Internal(ls(envir, all.names, sorted))
##   if (!missing(pattern)) {
##     if ((l1 <- length(grep("[", pattern, fixed = TRUE))) &&
##       l1 != length(grep("]", pattern, fixed = TRUE))) {
##       if (pattern == "[") {
```


Vous pouvez utiliser `rm` pour supprimer des objets dont vous n'avez plus besoin:

```
rm (x)
```

Si vous avez beaucoup de choses dans votre environnement et souhaitez les supprimer toutes, vous pouvez transmettre les résultats de `ls` à la fonction `rm`:

```
rm(list=ls())
```

Dans ce cas, nous avons spécifié que les résultats de `ls` devraient être utilisés pour le L'argument `list` dans `rm`. Lorsque vous attribuez des valeurs aux arguments par nom, vous *devez* utilisez l'opérateur `=` !!

Si au lieu de cela nous utilisons `<-`, il y aura des effets secondaires inattendus, ou vous pourriez avoir un message d'erreur:

```
rm(list <- ls ())
```

```
## Error in rm(list <- ls()): ... must contain names or character strings
```

- Quand R fait quelque chose d'inattendu! Les erreurs, comme ci-dessus, sont émises lorsque R ne peut pas procéder à un calcul.
- En revanche, les avertissements signifient généralement que la fonction a été exécutée, mais cela n'a probablement pas fonctionné comme prévu.
- Dans les deux cas, le message que R imprime vous donne généralement des indices sur la manière de résoudre un problème.

Il est possible d'ajouter des fonctions à R en écrivant un paquet, ou par obtenir un paquet écrit par quelqu'un d'autre. Au moment de l'écriture, il y a sont plus de 7 000 paquets disponibles sur CRAN (l'archive complète de R réseau). R et RStudio ont des fonctionnalités pour gérer les paquets:

- Vous pouvez voir quels paquets sont installés en tapant

```
installed.packages ()
```

- Vous pouvez installer des paquets en tapant

```
install.packages (" packagename "), où packagename est le nom du package, entre guillemets.
```

- Vous pouvez mettre à jour les paquets installés en tapant `update.packages ()`
- Vous pouvez supprimer un paquet avec `remove.packages (" packagename ")`
- Vous pouvez rendre un paquet disponible pour être utilisé avec `library (packagename)`

- ➊ Nommer toutes les panels de R studio
- ➋ Réaliser une addition et une multiplication dans la console
- ➌ Sauver un script R avec votre code d'addition

Section 8

Chercher de l'aide

R, et chaque paquet, fournissent des fichiers d'aide pour les fonctions. Pour chercher de l'aide sur une fonction d'une fonction spécifique qui est dans un package chargé

```
?nom_fonction  
help(nom_fonction)
```

Cela chargera une page d'aide dans RStudio (ou du texte brut dans R seul).

Pour demander de l'aide sur des opérateurs spéciaux, utilisez des guillemets:

```
? "+"
```

De nombreux paquets contiennent des “vignettes”: des tutoriels et des exemples de documentation. Sans aucun argument, `vignette()` listera toutes les vignettes pour tous les paquets installés;

```
vignette (package =" nom-du-paquet ")
```

listera toutes les vignettes disponibles pour

```
package-name et vignette ("vignette-name")
```

ouvriront la vignette spécifiée.

Si un paquet ne contient aucune vignette, vous pouvez généralement trouver de l'aide en tapant

```
help("nom-du-paquet").
```


Quand vous vous souvenez de la fonction

Si vous ne savez pas exactement dans quel paquet est une fonction ou comment elle est spécifiquement orthographiée, vous pouvez effectuer une recherche floue:

```
??nom_fonction
```

Section 9

Mélanger code et texte avec knitr

Principe de Claerbout (Géophysicien, Stanford)

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

knitr

pour générer des rapports qui combinent le texte, le code et les résultats.

Markdown

pour mettre en forme le texte

Markdown est un langage de balisage léger créé par John Gruber en 2004. Son but est d'offrir une syntaxe facile à lire et à écrire. Un document balisé par Markdown peut être lu en l'état sans donner l'impression d'avoir été balisé ou formaté par des instructions particulières. — Wikipedia

[Lien Wikipedia](#)

Code Chunks

code dans des blocs délimités par des guillemets triples suivis de `{r}`.

Rmarkdown est un univers extensible, si vous voulez continuer, lisez

<https://rmarkdown.rstudio.com/>

Dans la console

```
install.packages('knitr')
```


Par le menu


Tools -> Install Packages


Créer un de type fichier Rmarkdown (.Rmd)


Dans R Studio, cliquez sur Fichier → Nouveau fichier → R Markdown et vous obtiendrez une boîte de dialogue du type

New R Markdown

 Document

 Presentation

 Shiny

 From Template

Title:

Author:

Default Output Format:

☒ **HTML**
Recommended format for authoring (you can switch to PDF or Word output anytime).

☐ **PDF**
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

☐ **Word**
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

Vous créez un fichier avec une entête dite yaml du type

```
---  
title: "Initial R Markdown document"  
author: "Karl Broman"  
date: "April 23, 2015"  
output: html_document  
---
```

qui précise comment peut être transformé le fichier

Markdown est un langage à balise

- écrivez **en gras** en utilisant deux astérisques, comme ceci: ****gras****,
- écrivez en *italics* en utilisant des traits de soulignement, comme ceci: _italics_.

Vous pouvez créer une liste à puces en écrivant une liste avec des tirets ou astérisques, comme ceci:

- * gras avec double astérisque
- * italiques avec des soulignés
- * police de type code avec backticks

ou comme ça:

- gras avec double astérisque
- italiques avec des soulignés
- police de type code avec backticks

Vous pouvez créer une liste numérotée en utilisant simplement des chiffres. Vous pouvez utiliser le même nombre encore et encore si vous voulez:

- 1. gras avec double astérisque
- 1. italiques avec des soulignés
- 1. police de type code avec backticks

Cela apparaîtra comme:

- ❶ gras avec double astérisque
- ❷ italiques avec des soulignés
- ❸ police de type code avec backticks

Vous pouvez créer des en-têtes de section de différentes tailles en initiant une ligne avec un certain nombre de symboles #:

Titre

Section principale

Sous-section

Sous-sous-section

Vous *compilez* le document R Markdown en cliquant sur le “Knit HTML” en haut à gauche.

Un peu plus de Markdown

- Vous pouvez créer un hyperlien comme celui-ci:

[texte à afficher] (<http://the-web-page.com>).

- Vous pouvez inclure un fichier image comme ceci:

![Caption] (<http://url/for/file>)

Vous pouvez faire des indices (par exemple, F_2) avec `F~2~` et des exposants (par exemple, F^2) avec `F^2^`.

Si vous savez écrire des équations dans [LaTeX] (<http://www.latex-project.org/>), vous serez heureux de savoir que vous pouvez utiliser `$ $` et `$$ $$` pour insérer des équations mathématiques, comme `$E = mc^2$` $E = mc^2$ et

`$$y = \mu + \sum_{i=1}^p \beta_i x_i + \epsilon$$`

$$y = \mu + \sum_{i=1}^p \beta_i x_i + \epsilon$$

Markdown est intéressant et utile, mais le plus grand intérêt vient du mélange du texte balisé avec des morceaux de code R.

- Quand traité, le code R sera exécuté; s'ils produisent des valeurs, figures, ceux-ci seront insérés dans le document final.

Les morceaux de code ressemblent à ceci:

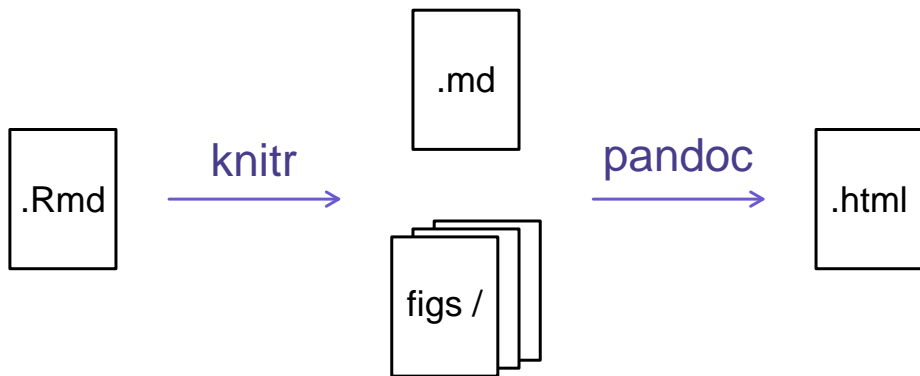
```
```{r load_data}  
gapminder <- read.csv("~/Desktop/gapminder.csv")
```
```

C'est une bonne idée de donner chaque morceau un nom, car ils vous aideront à corriger les erreurs et, si des graphiques sont produit, les noms de fichiers sont basés sur le nom du bloc de code les a produites.

Lorsque vous appuyez sur le bouton “Knit HTML”, le document R Markdown est traité par [knitr] (<http://yihui.name/knitr>) et un simple Markdown document est produit (ainsi que, potentiellement, un ensemble de fichiers de figure): le code R est exécuté et remplacé à la fois par l'entrée et la sortie; si les chiffres sont produits, des liens vers ces chiffres sont inclus.

Les documents Markdown et figure sont ensuite traités par l'outil [pandoc] (<http://pandoc.org/>), qui convertit le fichier Markdown en fichier fichier html, avec les chiffres incorporés.

Comment les choses sont compilées II



Il y a une variété d'options pour affecter la façon dont les morceaux de code sont traités.

- Utilisez `echo = FALSE` pour éviter que le code lui-même ne soit affiché.
- Utilisez `results = "hide"` pour éviter d'imprimer des résultats.
- Utilisez `eval = FALSE` pour que le code soit affiché mais pas évalué.
- Utilisez `warning = FALSE` et `message = FALSE` pour masquer les avertissements ou messages produits.
- Utilisez `fig.height` et `fig.width` pour contrôler la taille des figures produit (en pouces).

- Knitr in a knutshell tutorial
- Dynamic Documents with R and knitr (book)
- R Markdown documentation
- R Markdown cheat sheet

- 1 Créer un fichier Rmarkdown qui produira une sortie html avec un code chunk de votre choix