Initiation à la programmation sur R Studio

Quentin LAJAUNIE

quentin_lajaunie@hotmail.fr

Présentation du cours

Objectifs :

- Se familiariser avec le logiciel et revoir les principes algorithmiques de base
- Importer des données, et faire face aux données manquantes
- Créer un indice composite à partir de ces données :
  - Normalisation des données et pondération
  - Méthodes statistiques économétriques
  - Analyse en Composantes Principales

Notation :

- Rendu des TDs et participation
- Projet ou partiel

Chapitre 1

Concepts algorithmiques de base

Introduction

Un algorithme est une séquence d'instructions programmées qui peut être exécutée par un ordinateur. Les manipulations décrites ci-dessus ne sont pas représentatives d'un travail de programmation. Pour pouvoir développer un programme **R** en vue d'un traitement statistique étendu, il faut non seulement connaître la syntaxe spécifique à **R** mais aussi (et surtout) avoir une connaissance de base des algorithmes.

Le chapitre suivant a pour but de présenter ces connaissances de base.

Sommaire

Concepts algorithmiques de base

01	LES VARIABLES	6
02	LES VECTEURS	8
03	LES MATRICES	12
04	LISTE ET DATAFRAME	16
05	LA CONDITION IF	17
06	LA STRUCTURE ITÉRATIVE	21
07	LES FONCTIONS	30
08	EXERCICES	33

Les variables
(1/2)

Les variables sont des emplacements réservés dans la mémoire d'un ordinateur qui stockent des valeurs. Les variables les plus simples sont utilisées pour stocker des valeurs numériques. Par la suite, nous verrons que nous pouvons également stocker des structures plus complexes, telles que des vecteurs, des matrices et des tableaux. La création d'une variable comporte deux étapes : la déclaration et l'affectation.

Nous pouvons considérer quatre types de variables dans R :

- numériques / quantitatives ;
- facteurs / qualitatives avec un nombre de modalités fini ;
- booléennes (True / False) ;
- textes / caractères contenant des chaînes de caractères.

Les variables
(2/2)

Pour connaître le type d'une variable donnée, nous pouvons utiliser la fonction **class** :

```
1  a <- "bonjour"
2  class(a)
3
4  b <- TRUE
5  class(b)
6
7  c <- pi
8  class(c)
```

Les vecteurs
(1/4)

Un vecteur est une structure de données de base dans R qui contient un ou plusieurs éléments du même type. Un vecteur est défini par les types de ses éléments et sa longueur. Les types de données peuvent être integer, double, character, etc. et peuvent être vérifiés avec la fonction **typeof**(). La longueur d'un vecteur correspond au nombre d'éléments qu'il contient. Il se calcule avec la fonction **length**().

```
1  # creation du vecteur
2  v_My_Vector1 <- vector(mode = "numeric" , length = 10)
3  print(v_My_Vector1) # affichage
4
5  # attribution des valeurs au vecteur
6  v_My_Vector1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
7  print(v_My_Vector1) # affichage
8
9  # taille du vecteur
10 length(v_My_Vector1)
11
12 # creation du vecteur
13 v_My_Vector2 <- vector(mode = "numeric" , length = 10)
14 v_My_Vector2 <- rep(3, 10)
15 print(v_My_Vector2) # affichage
```

```r

# creation du vecteur
v_My_Vector1_2 <- vector(mode = "numeric" , length = 20)
v_My_Vector1_2 <- c(v_My_Vector1, v_My_Vector2)
print(v_My_Vector1_2) # affichage

# creation du vecteur
v_My_Vector3 <- vector(mode = "numeric" , length = 10)
v_My_Vector3 <- seq(10, 1)
print(v_My_Vector3) # affichage

v_My_Vector3[2:3] # affichage des elements 2 et 3 du vecteur
v_My_Vector3[4] # affichage du quatrieme element du vecteur

# creation du vecteur
v_My_Vector4 <- vector(mode = "numeric" , length = 10)
set.seed(1)
v_My_Vector4 <-  sample(1:10,10)
print(v_My_Vector4)

# creation du vecteur
v_My_Vector5 <- vector(mode = "numeric" , length = 10)
set.seed(1)
v_My_Vector5 <- sample(1:10,10,replace = T)
print(v_My_Vector5)

# declaration d un vecteur boolean
v_My_Vector_Boolean <- logical(length = 3)

# attribution des valeurs au vecteur
v_My_Vector_Boolean <-c(TRUE, FALSE, TRUE)
print(v_My_Vector_Boolean)
```

Les vecteurs
(3/4)

Des fonctions comme la moyenne, maximum, minimum, etc... peuvent ensuite être appliquées sur les vecteurs :

```r
# creation du vecteur
v_My_Vector <- vector(mode = "numeric" , length = 10)
set.seed(1)
v_My_Vector <- sample(5:30,10,replace = T)
print(v_My_Vector)

max(v_My_Vector) # valeur max du vecteur
min(v_My_Vector) # valeur min du vecteur
mean(v_My_Vector) # valeur moyenne du vecteur
sum(v_My_Vector) # somme des termes du vecteur
```

Les vecteurs
(4/4)

Des opérations peuvent également être faites sur des vecteurs de même taille :

```
1  # creation des vecteurs
2  v_My_Vector1 <- vector(mode = "numeric" , length = 10)
3  v_My_Vector2 <- vector(mode = "numeric" , length = 10)
4
5  # Association de jeux de donnees aux vecteurs
6  v_My_Vector1 <- seq(0,9)
7  v_My_Vector2 <- seq(9,0)
8
9  # creation du vecteur operation
10 v_My_VectorOperation <- vector(mode = "numeric" , length = 10)
11
12 v_My_VectorOperation <- v_My_Vector1+v_My_Vector2 # addition des termes
13 v_My_VectorOperation <- v_My_Vector1-v_My_Vector2 # soustraction des termes
14
15
16 v_My_VectorOperation <- v_My_Vector1 * v_My_Vector2 # multiplaction des termes
17 sum(v_My_VectorOperation) # somme des termes apres multiplication
```

Les matrices
(1/4)

Une matrice (ou, plus généralement, un tableau, surtout si le nombre de dimensions est supérieur à 2) est une structure de données à deux dimensions dans le langage de programmation R. Elle est similaire à un vecteur, mais elle contient également l'attribut dimensionnel. Les dimensions peuvent être vérifiées directement à l'aide de la fonction attributes() :

```
1  # creation de la matrice en declarant uniquement les lignes
2  m_My_Matrix_1 <- matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 2)
3  print(m_My_Matrix_1) # affichage
4
5  # creation de la matrice en declarant uniquement les colonnes
6  m_My_Matrix_2 <- matrix(data = c(1, 2, 3, 4, 5, 6), ncol = 3)
7  print(m_My_Matrix_2) # affichage
8
9  length(m_My_Matrix_2) # calcul du nombre d elements de la matrice
10 ncol(m_My_Matrix_2) # nombre de colonnes
11 nrow(m_My_Matrix_2) # nombre de lignes
12
13 t(m_My_Matrix_2) # transposition de la matrice
14
15 # creation de la matrice en declarant les lignes et les colonnes
```

Les matrices
(2/4)

Autres exemples :

```
16  m_My_Matrix_3 <- matrix(data = 0, nrow=3, ncol = 3)
17  print(m_My_Matrix_3)
18
19  # creation de la matrice
20  m_My_Matrix_4 <-matrix(data = seq(1,4), nrow=2, ncol = 2)
21
22  # inversion de la matrice m_My_Matrix_4 et stockage dans m_My_Matrix_5
23  m_My_Matrix_5 <- solve(m_My_Matrix_4)
24
25  # creation de la matrice en declarant les lignes et les colonnes
26  m_My_Matrix_6 <- matrix(data = diag(3), nrow=3, ncol = 3)
27  print(m_My_Matrix_6)
```

Les matrices
(3/4)

De même que pour les vecteurs, les fonctions comme la moyenne, maximum, minimum, etc... peuvent ensuite être appliquées. Les opérations fonctionnent également, et permettent de faire des sommes de matrices, ou encore des produits matriciels. Ces opérations se font conditionnellement à la taille des matrices.
Par exemple, dans l'application suivante, la console de R enverra le message suivant: "*Error in m_My_Matrix_1 + m_My_Matrix_2 : tableaux de tailles inadéquates*"

```
1  m_My_Matrix_1 <- matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 2)
2  print(m_My_Matrix_1)
3
4  m_My_Matrix_2 <- matrix(data = c(1, 2, 3, 4, 5, 6), ncol = 2)
5  print(m_My_Matrix_2)
6
7  m_My_Matrix_1 + m_My_Matrix_2
```

Les matrices
(4/4)

Quelques exemples d'applications :

```
1
2   # creation de la matrice avec des chiffres aleatoires
3   set.seed(1)
4   m_My_Matrix <- matrix(data =  sample(1:10,3,replace=T), nrow = 2, ncol=3)
5   print(m_My_Matrix) # affichage
6
7   sum(m_My_Matrix) # somme des elements de la matrice
8   mean(m_My_Matrix) # moyenne des elements de la matrice
9
10  # creation de la matrice
11  m_My_Matrix_2 <- matrix(data =  rep(2), nrow = 2, ncol=3)
12  print(m_My_Matrix_2) # affichage
13
14  # creation de la matrice
15  m_My_Matrix_3 <- matrix(data = seq(1,6), nrow = 2, ncol = 3)
16  print(m_My_Matrix_3) # affichage
17
18  m_My_Matrix_2 + m_My_Matrix_3 # addition
19  m_My_Matrix_2 - m_My_Matrix_3 # soustraction
20
21  # creation des matrices
22  m_My_Matrix_4 <- matrix(data = seq(1,4), nrow=2, ncol = 2)
23  m_My_Matrix_5 <- matrix(data = 0, nrow=2, ncol = 2)
24
25  m_My_Matrix_5 <- solve(m_My_Matrix_4) # inversion de la matrice 4
26
27  m_My_Matrix_5 * m_My_Matrix_4 # multiplication des termes de chaque matrice
```

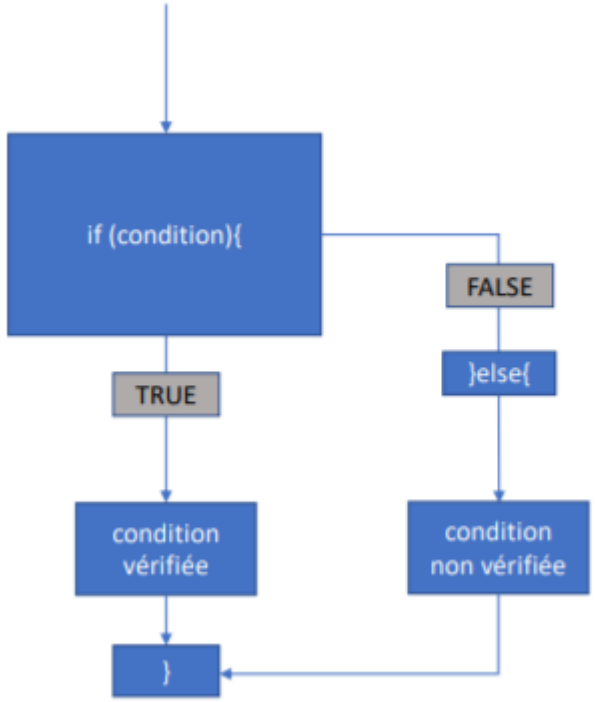
Liste et dataframe

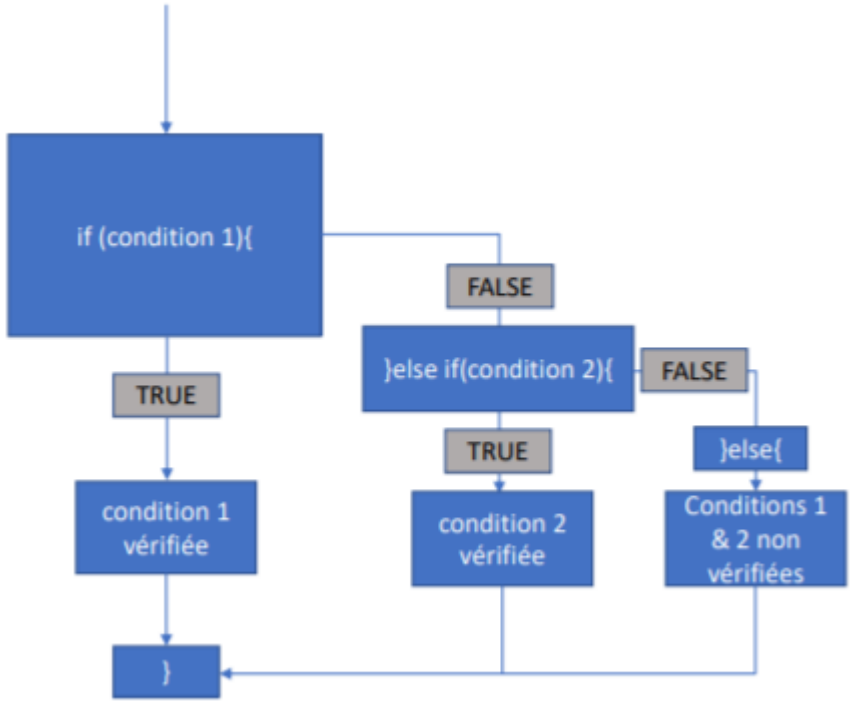
Un vecteur avec tous les éléments du même type est appelé un vecteur atomique, mais un vecteur avec des éléments de types différents est appelé une liste. Un dataframe est une structure de données bidimensionnelle dans **R**. C'est un cas particulier de liste, dans laquelle chaque élément a la même longueur, mais ou les éléments peuvent être de types différents.

```
1  # Creation de la liste
2  l_My_List <- list('a' = 3.14, 'b' = TRUE, 'c' = 1:10, 'd' = '21/10/2019')
3  l_My_List # affichage
4
5  # Creation du dataframe
6  df_My_DataFrame <- data.frame('a' = 1:2, 'b' = c(21,15), 'd' = c('M1','M2'))
7  df_My_DataFrame # affichage
```

La condition if
(1/4)

Les déclarations conditionnelles permettent d'effectuer différentes actions selon qu'une condition est évaluée comme étant vraie ou fausse.





La condition if
(2/4)

```r
# tirage aleatoire d un nombre entre 1 et 10
Random_variable <-sample(1:10,1)

# condition pour savoir si le tirage est pair ou impair
if(Random_variable modulo 2==0){
  results <- "pair"
}else{
  results <- "impair"
}
results
```

La condition if peut être utilisée dans le cadre d'une variable boolean: si le résultat est égale à ... alors "A" , sinon "B" . Le résultat aura donc deux sorties possibles. L'exemple précédent permet de savoir si le nombre aléatoire est pair ou impair. Toutefois, cette condition peut être étendue à une multitude de cas, avec plus de deux sorties.

La condition if
(3/4)

```r
# tirage aleatoire d un nombre entre 1 et 150
Random_variable <-sample(1:150,1)

# condition pour savoir si le resultat est petit, moyen ou grand
if(Random_variable >= 1 && Random_variable < 50)
{
  results <- "petit"
}else if(Random_variable >= 50 && Random_variable < 100){
  results <- "moyen"
}else{
  results <- "grand"
}
results
```

La condition if
(4/4)

Enfin, plusieurs conditions peuvent être imbriquées. Ainsi, une première condition peut contenir une seconde condition, comme suit:

```
# tirage aleatoire d un nombre entre 1 et 150
Random_variable <-sample(1:150,1)

# condition pour savoir si le resultat est petit, moyen ou grand, et pair ou impair
if(Random_variable >= 1 && Random_variable < 50)
{
  if(Random_variable modulo 2==0){
    results <- "petit et pair"
  }else{
    results <- "petit et impair"
  }
}else if(Random_variable >= 50 && Random_variable < 100){
  if(Random_variable modulo 2==0){
    results <- "moyen et pair"
  }else{
    results <- "moyen et impair"
  }
}else{
  if(Random_variable modulo 2==0){
    results <- "grand et pair"
  }else{
    results <- "grand et impair"
  }
}
```

La structure itérative

La structure itérative permet, sous la forme de boucles, de répéter un bloc de code spécifique. Cette structure peut prendre la forme d'une boucle for, ou d'une boucle while. La boucle for sera souvent utilisée pour répéter une opération un nombre de fois, tandis que la boucle while sera utilisée jusqu'à ce qu'une condition ne soit plus respectée.

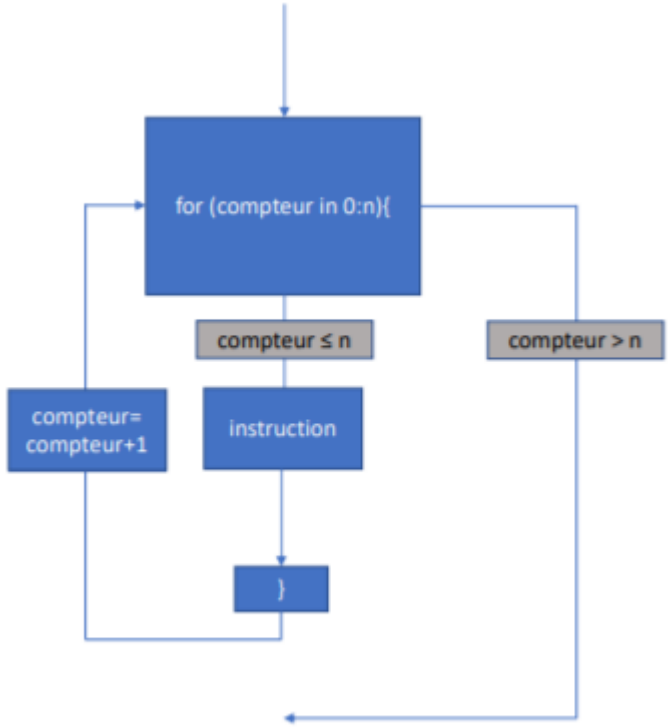
- **for** : Une boucle for permet de répéter l'exécution d'une séquence d'instructions. Elle permet d'itérer (avec un curseur défini).

- **while** : La boucle while intègre une condition qui contraint le programme à continuer des itérations jusqu'à ce que cette condition soit respectée. La boucle continue "tant que...".

Une des différences majeure avec la boucle **for** réside dans l'incrémentation du compteur. Dans la boucle **for**, le compteur est incrémentée à chaque itération. Dans la boucle **while**, il faut obligatoirement ajouter une ligne permettant de passer à l'itération suivante.

La structure itérative : la boucle for
(1/4)

Pour faire une boucle for, il faut initialiser un certain nombre de paramètres, notamment le nombre d'itérations choisi. Il peut également être important d'initialiser le compteur, mais ce n’est pas systématique. Enfin, l'indentation du code, les commentaires, et le nom des variables permet de limiter les erreurs et facilite la relecture.



La structure itérative : la boucle for
(2/4)

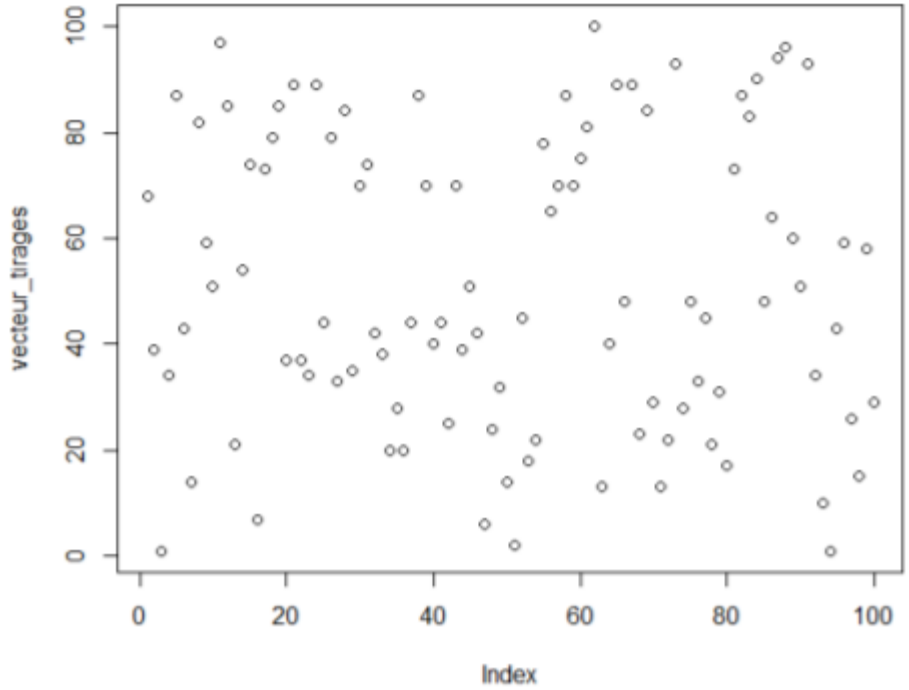
Dans l'exemple ci-dessous on cherche à calculer la moyenne de plusieurs tirages aléatoires. On définit le nombre de tirages, et la somme des tirages dans laquelle on stockera à chaque itération la valeur tirée. La boucle permet de réaliser le nombre de tirage choisi. Enfin, une fois les itérations effectuées, on calcule la valeur moyenne en divisant la somme par le nombre de tirages.

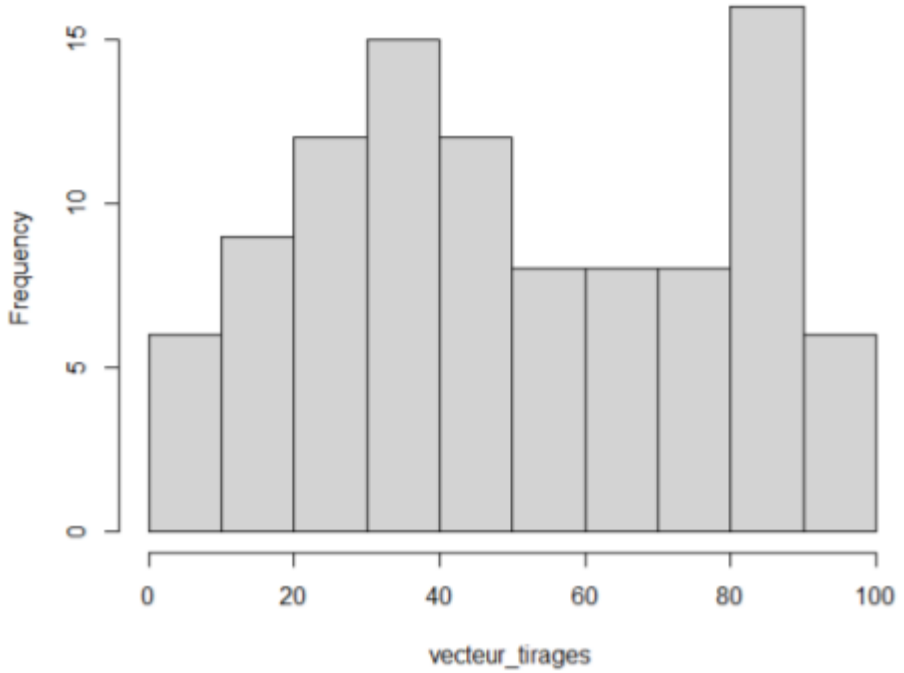
```
1
2  set.seed(1)
3
4  # initialisation des parametres
5  compteur_tirage <- 1 # compteur tirage
6  somme_tirages <- 0 # somme des tirages
7  Nb_tirages <- 100 # nombre de tirages
8
9
10 for (compteur_tirage in 1:Nb_tirages){
11   # tirage al atoire entre 1 et 100
12   valeur_tirage <- sample(1:100,1)
13
14   # calcul de la somme (processus iteratif)
15   somme_tirages <- somme_tirages + valeur_tirage
16 }
17
18 # calcul de la moyenne des tirages
19 moyenne_tirage <- somme_tirage / Nb_tirages
```

La structure itérative : la boucle for
(3/4)

L'avantage du stockage vectoriel est multiple. Il permet de faire référence à une itération spécifique (un tirage spécifique ici) une fois la boucle réalisée. D'autre part, il permet une représentation graphique pour faciliter la visualisation des résultats. La commande **plot** ne sera pas forcément la plus adaptée contrairement à un histogramme (commande **hist**).

```
1 plot(vecteur_tirages)
2 hist(vecteur_tirages)
```





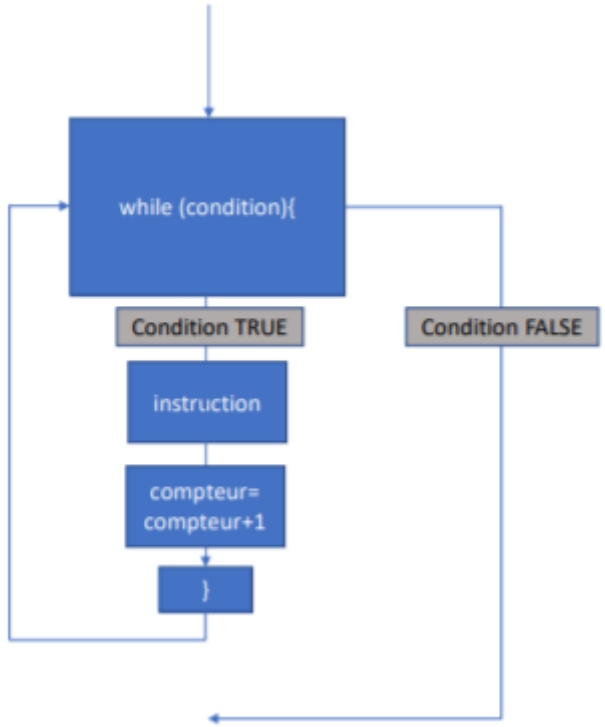
La structure itérative : la boucle for
(4/4)

Enfin, il est important de noter que la condition **if** peut être utilisée à l'interieur d'une boucle for. Si vous souhaitez par exemple avoir des tirages aléatoires, mais que vous ne souhaitez aucune valeur inférieure à 20, vous pouvez ajouter une condition pour chaque itération.

```
1
2  for (compteur_tirage in 1:Nb_tirages){
3    # tirage aleatoire entre 1 et 100
4    valeur_tirage <- sample(1:100,1)
5
6    # recuperation du tirage dans le vecteur si > 20
7    if (valeur_tirage>20){
8      vecteur_tirages[compteur_tirage] <- valeur_tirage
9    }
10 }
```

La structure itérative : la boucle while
(1/4)

La boucle while intègre une condition qui contraint le programme à continuer des itérations jusqu'à ce que cette condition soit respectée. La boucle continue "tant que...". Une des différences majeure avec la boucle **for** réside dans l'incrémentation du compteur. Dans la boucle **for**, le compteur est incrémentée à chaque itération. Dans la boucle **while**, il faut obligatoirement ajouter une ligne permettant de passer à l'itération suivante.



La structure itérative : la boucle while
(2/4)

En prenant le dernier exemple de la sous-section précédente, nous avions intégré une condition pour ne stocker que les tirages ayant une valeur supérieure à 20 dans le vecteur "vecteur_tirages". Le vecteur ayant été initialisé comme un vecteur nul, le résultat permet d'obtenir un vecteur contenant des 0 lorsque les tirages sont inférieurs à 20, et contenant les valeurs tirées sinon. L'utilisation de la boucle while permet d'obtenir un vecteur ne contenant que des tirages supérieurs à 20.

La structure itérative : la boucle while
(3/4)

```
1  set.seed(1)
2
3  # initialisation des parametres
4  compteur_tirage <- 1 # compteur tirage
5  vecteur_tirages <- 0 # somme des tirages
6  Nb_tirages <- 100 # nombre de tirages
7  vecteur_tirages <- as.vector(rep(0,Nb_tirages)) # vecteur des tirages
8
9  while(vecteur_tirages[Nb_tirages]==0)
10   {
11   # tirage aleatoire entre 1 et 100
12   valeur_tirage <- sample(1:100,1)
13   if (valeur_tirage>20){
14     # r cup ration du tirage dans le vecteur
15     vecteur_tirages[compteur_tirage] <- valeur_tirage
16     compteur_tirage <- compteur_tirage + 1
17     }
18 }
19
20 # calcul de la somme des tirages
21 somme_tirage <- sum(vecteur_tirages)
22
23 # calcul de la moyenne des tirages
24 moyenne_tirage <- somme_tirage / Nb_tirages
```

La structure itérative : la boucle while
(4/4)

Une autre utilisation de la boucle while peut être aussi de réaliser une séquence d'itérations jusqu'à ce qu'une valeur souhaitée soit obtenue. Par exemple faire un tirage aléatoire jusqu'à ce que le résultat soit supérieur à 80.

```
1  set.seed(1)
2
3  # initialisation des parametres
4  compteur_tirage <- 0 # compteur tirage
5  vecteur_tirages <- 0 # somme des tirages
6  Nb_tirages <- 100 # nombre de tirages
7
8  valeur_tirage<-0
9  while(valeur_tirage<80)
10   {
11   # tirage al atoire entre 1 et 100
12   valeur_tirage <- sample(1:100,1)
13   compteur_tirage <- compteur_tirage + 1
14 }
15
16 # Affichage du resultat
17 compteur_tirage
```

Les fonctions
(1/3)

Nous avons vu jusqu'à présent plusieurs fonctions de base permettant de faire des opérations, des calculs vectoriels et des calculs matriciels. Nous avons également vu plusieurs principes algorithmiques permettant d'intégrer une condition, ou encore de réaliser une séquence d'instructions à partir d'un processus itératif. L'ensemble de ces applications ont été illustrées par des exemples à l'intérieur d'un script.

Une nouvelle notion importante en programmation est celle de la fonction. Presque tout dans la programmation est fait à l'aide de fonctions. Une fonction est un morceau de code écrit pour exécuter une tâche spécifique. Elle peut (ou non) accepter des arguments ou des paramètres, et elle peut (ou non !) renvoyer une ou plusieurs valeurs.

```
1  function(argument_list){
2  body
3  }
```

Les fonctions
(2/3)

Le code qui se trouve entre les accolades est le corps de la fonction. Notez que si vous utilisez des fonctions intégrées, vous n'avez qu'à vous préoccuper de communiquer efficacement les arguments d'entrée corrects (argument_list) et de gérer la ou les valeurs de retour, le cas échéant, en utilisant **return**() avant la dernière accolade.

Une fonction doit être associée à un nom de variable pour pouvoir ensuite être appelée. La fonction doit ensuite être compilée une première fois avant de pouvoir être utilisée dans un script. Par exemple, si nous souhaitons faire une fonction qui permet de convertir une température en kelvin qui était initialement en fahrenheit, nous pouvons réaliser la séquence suivante :

```
1  # fonction de conversion fahrenheit en kelvin
2  convert_fahrenheit_to_kelvin <- function(temp = temperature) {
3    # calcul pour la conversion
4    kelvin <- ((temp - 32) * (5 / 9)) + 273.15
5
6    # sortie de la fonction
7    return(kelvin)
8  }
```

Les fonctions
(3/3)

Une fois cette fonction réalisée, l'utilisateur doit d'abord la compiler. Ensuite, il peut l'appeler dans un script en utilisant le nom associé.

```
# Appel de la fonction
convert_fahrenheit_to_kelvin(32)
```

Exercices

Concepts algorithmiques de base

Exercice 1 :

Créez un vecteur *V1* de nombres aléatoires comprises entre -10 et 10 avec remise, et de taille *L=100*.
Déterminez ensuite en une ligne le nombre de valeurs inférieures à 0 et le nombre de valeurs supérieures ou égales à 0.
Enfin, modifiez ce vecteur de sorte que les valeurs positives ou nulles prennent la valeur de 1 et les valeurs négatives prennent la valeur de 0.

Exercice 2 :

Créez un vecteur *V2* de taille 10, dans lequel les valeurs sont égales à *V2=[0.5, 1, 1.5, ..., 4.5, 5]*.
Aide : seq(valeur_min, valeur_max, by = pas)

Exercice 3 :

Créez une matrice *M1* nulle de dimensions $[10 \times 10]$.
Modifiez la matrice *M1* de sorte que *M1* soit une matrice identité. Cette modification doit être opérer itérativement.

Exercice 4 :

Créez une matrice *M2* de nombres aléatoires de dimensions $[100 \times 100]$.
Modifiez la matrice *M2* de sorte que *M2* soit une matrice triangulaire inférieure.

Exercices

Concepts algorithmiques de base

Exercice 5 :

Dans un vecteur, stockez 500 tirages aléatoires d'une loi de Poisson de paramètre $\lambda = 3$. Une fois le tirage effectué, faites une représentation graphique du résultat en utilisant un graphique adapté.

Aide : La fonction **rnorm** vue en cours utilise le préfixe **r** pour générer des nombres aléatoires (r pour randomize). La loi de poisson s'écrit **pois**.

Exercice 6 :

Utilisez la fonction permettant d'obtenir le quantile théorique de la loi de Poisson associé au chiffre 2, en considérant le même paramètre $\lambda = 3$. A quoi correspond le quantile obtenu?

Ensuite, utilisez le vecteur obtenu dans l'exercice précédent, et vérifiez empiriquement la cohérence avec le quantile théorique. Pour ce faire, vous pouvez utiliser la condition **if**, ainsi qu'un processus itératif.

Aide : Le préfixe **r** permet de générer des nombres aléatoires. Le préfixe **q** est utilisé pour obtenir le quantile.

Exercice 7 :

En utilisant un processus itératif, calculez 10!.

Aide : pour rappel, une fonction factorielle se calcule comme suit:

$$n! = \prod_{1 \leq i \leq n} i = 1 \times 2 \times \cdots \times (n-1) \times (n)$$

Exercices

Concepts algorithmiques de base

Exercice 8 :

En reprenant une structure algorithmique similaire à celle de l'exercice précédent, développez une fonction qui vous permet de calculer la fonction factorielle.

Exercice 9 :

Ecrire une fonction qui prend comme input un vecteur de taille quelconque, et qui renvoie en sortie le chiffre le plus petit. La fonction **min** ne doit pas être utilisée.

Exercice 10 :

Ecrire une fonction qui prend comme input un nombre quelconque et qui renvoie en TRUE si ce nombre est premier, et FALSE sinon.

Aide : Un nombre premier admet exactement 2 diviseurs.

Chapitre 2

Indicateur composite – Définition et concept

Sommaire

Indicateur composite – Définition et concept

01 INTRODUCTION 38

02 LES ÉTAPES DE CONSTRUCTION 41

03 EXEMPLE D’INDICATEUR COMPOSITE 44

04 CADRE THÉORIQUE 45

Introduction
(1/3)

Les indicateurs composites sont des indicateurs qui permettent de fournir des comparaisons entre des pays, entre des entreprises, ou entre des individus. Ils permettent d'illustrer des questions complexes dans des domaines variés, par exemple l'environnement, l'économie, la société ou le développement technologique.

Les indicateurs composites (index) sont plus faciles à interpréter. Mais il convient de faire attention à leur construction pour éviter que les conclusions ne soient trompeuses.

Introduction
(2/3)

Définition : *« Un indicateur est une mesure quantitative ou qualitative dérivée d'une série de faits observés qui peut révéler les positions relatives (par exemple d'un pays) dans un domaine donné. Lorsqu'il est évalué à intervalles réguliers, un indicateur peut mettre en évidence la direction du changement dans différentes unités et dans le temps. »* (OECD 2008)

Il est construit à partir d'indicateurs individuels compilés en un seul indice et permet de mesurer des concepts multidimensionnels (par exemple la compétitivité, l'industrialisation, la durabilité, les notes ESG, la performance d'une entreprise). Un indicateur est construit généralement avec plusieurs variables quantitatives et/ou qualitatives agrégées.

Introduction
(3/3)

Avantages : Les indices composites permettent de résumer des réalités complexes et multidimensionnelles en vue d'aider les décideurs. Ils sont plus facilement interprétables, permettent d'évaluer les progrès des pays ou des entreprises dans le temps, et peuvent inclure de nombreuses informations.

Inconvénients : Si ils sont mal construits, ils peuvent être mal interprétés et peuvent entraîner des conclusions simplistes. La sélection des indicateurs et la pondération peuvent faire l'objet de conflit politique. Enfin, les indices composites peuvent masquer certaines défaillances, accroître la difficulté à identifier les bonnes mesures, et conduire à des actions inappropriées.

Les étapes de construction
(1/3)

Un indicateur composite peut être construit en suivant les 10 étapes décrites ci-dessous :

Etape 1 - Cadre théorique : permet d'offrir un cadre théorique aux indicateurs sélectionnés et à leur combinaison. Cette étape permet d'établir des critères de sélection, structurer des sous-groupes du phénomène, et obtenir une compréhension de ce qui est mesuré dans un cadre multidimensionnel.

Etape 2 - Sélection des données : vérification de la solidité des indicateurs, de leur mesurabilité, de leur couverture, de leur pertinence, et de leur relation entre eux. Il convient également de discuter des forces et des faiblesses de ces indicateurs.

Etape 3 - Imputation des données manquantes : approches statistiques permettant d'estimer les valeurs manquantes, et étudier les valeurs extrêmes pouvant être aberrantes.

Les étapes de construction
(2/3)

Etape 4 - Analyse multivariée : analyse exploratoire pour étudier la structure des indicateurs, l'adéquation des données, les choix méthodologiques de pondération et d'agrégation. Permet également d'identifier des groupes d'indicateurs et des groupes de pays similaires (aide à l'interprétation des résultats).

Etape 5 - Normalisation : pour rendre les données comparables. Attention à bien étudier les valeurs extrêmes.

Etape 6 - Pondération et agrégation : en fonction du cadre théorique. Attention au problème de corrélation entre les indicateurs. Plusieurs méthodes et procédures de normalisation peuvent être proposées.

Etape 7 - Robustesse et sensibilité : permet d'évaluer la robustesse d'indicateur avec des mécanismes d'inclusion ou d'exclusion de certains indicateurs.

Les étapes de construction
(3/3)

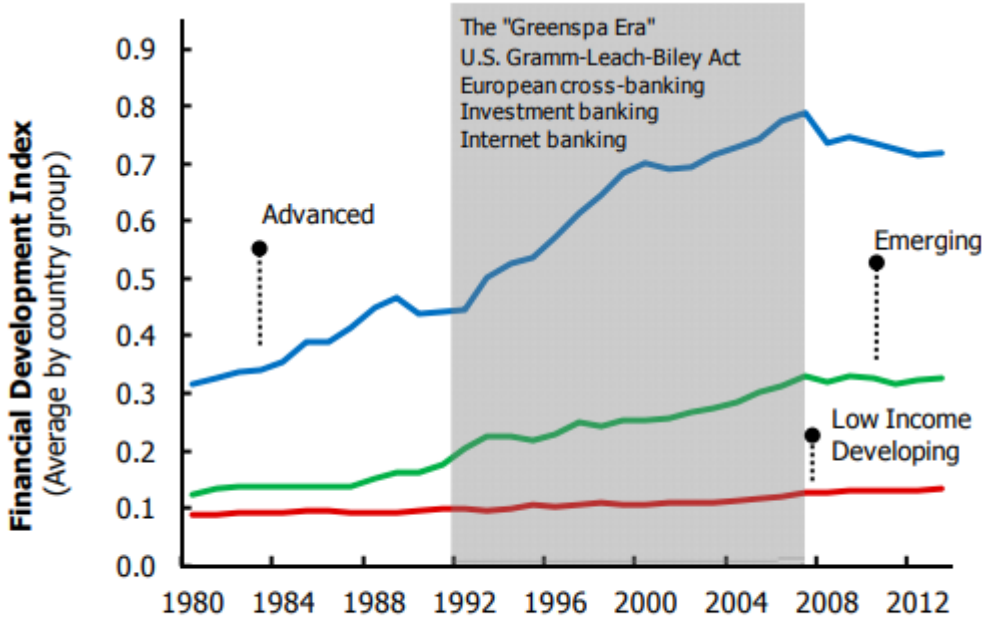
Etape 8 - Retour aux données réelles : les indicateurs et valeurs sous-jacentes doivent être transparents. Ensuite, il faut analyser les résultats pour voir la cohérence avec la réalité.

Etape 9 - Lien avec des variables : regarder si l'indicateur est corrélé avec d'autres indicateurs existants ou avec certaines variables utilisées pour approximer le fait étudié.

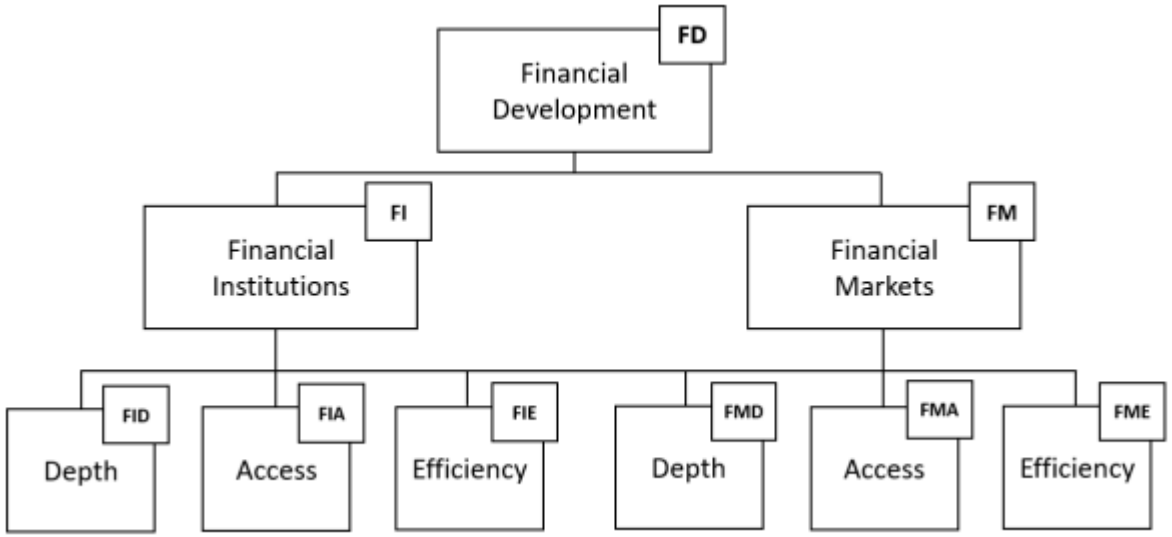
Etape 10 - Présentation et visualisation : présentation de l'indicateur de façon précise pour faciliter son utilisation et son interprétation.

Exemple d'indicateur composite

Un indicateur de développement financier s'appuie sur 6 sous-indicateurs (Cihak et al, 2012 ; Svirydzenka, 2016) :



Source: IMF staff estimates



Cadre théorique
(1/2)

« Ce qui est mal défini est susceptible d'être mal mesuré. » (OECD - 2008)

Le cadre théorique d'un indice composite s'appuie sur 3 piliers qui sont :

- La **définition du concept**. Cette définition doit permettre de comprendre clairement ce qui est mesuré et doit permettre de motiver que l'indice reflète une réalité économique, présentant un lien clair entre un fait et la mesure proposée. Lorsque la mesure fait l'objet de controverses, la qualité et la pertinence doivent être évaluées par les utilisateurs. Prenons l'exemple des scores ESG (Environnement, Social et Gouvernance) : les méthodologies permettant d'attribuer un score ESG aux entreprises sont nombreuses. Ces scores s'appuient généralement sur 3 sous-indices (E, S et G) calculés à partir de variables sous-jacentes. Mais il n'y a pas de consensus, et certaines méthodologies ou certains scores peuvent faire l'objet de controverses, en témoigne l'article *« The ESG Mirage »* de Bloomberg en décembre 2021. Les différents points de vue sont à l'origine de résultats parfois très divergents (Berg et al., 2019).

Cadre théorique
(2/2)

- Les **sous-groupes** correspondent généralement à des réalités multidimensionnels, comme la notation ESG qui s'appuie sur les dimensions E, S et G. D'autres indicateurs sont également créés avec cet aspect multidimensionnel, c'est notamment le cas de l'indice de développement financier de Svirydzenka (2016) qui utilise 6 sous-indicateurs. Ces sous-groupes ne sont pas nécessairement statistiquement indépendants les uns des autres, et les liens qui peuvent exister entre ces sous-groupes doivent être décrits et présentés.

- Les **critères de sélection** permettent de déterminer si un indicateur sous-jacent doit être inclus dans l'indice composite. Par exemple, un indice d'innovation pourrait combiner les dépenses de R&D et le nombre de nouveaux produits et services afin de mesurer l'ampleur de l'activité innovante d'un pays donné. Toutefois, seul le dernier ensemble d'indicateurs de sortie doit être inclus si l'indice est destiné à mesurer les performances en matière d'innovation.

Chapitre 3

Les données et la gestion des valeurs manquantes

Sommaire

Les données et la gestion des valeurs manquantes

01	CONCEPTS ET DÉFINITIONS	49
02	GESTION DES DONNÉES MANQUANTES	51
03	IMPUTATIONS SIMPLES	53
04	INTERPOLATION ET EXTRAPOLATION LINÉAIRE	56
05	RÉGRESSION LINÉAIRE	59
06	EXERCICES	63

Concepts et définitions
(1/2)

La qualité et la robustesse d'une analyse statistique ou empirique peut dépendre des données utilisées. Les valeurs manquantes peuvent être recalculées pour avoir des séries de données complètes. Les valeurs extrêmes doivent également être évaluées.

Considérer une valeur non observée comme manquante revient à considérer qu'il existe une valeur réelle sous-jacente qui aurait pu être observée. En revanche, certaines données manquantes n'ont pas systématiquement de variables réelles sous-jacentes. Prenons l'exemple d'un sondage pour le second tour des présidentielles : ne pas choisir l'un ou l'autre candidat en réponse à une enquête peut constituer une information qui permettrait d'identifier une troisième réponse "ne sait pas" ou "vote blanc".

Lorsque les données manquantes sont des valeurs non observées qui seraient significatives pour une analyse si elles étaient observées, il convient de leur imputer (attribuer) des valeurs.

Concepts et définitions
(2/2)

Il existe différents types de données manquantes (OECD, 2008) :

- **Manquantes complètement au hasard** *(Missing Completely At Random – MCAR)* :

  Les données manquantes ne dépendent pas d'autres variables telles que la variable d'intérêt ou d'autres variables observées dans l'ensemble de données.

- **Manquantes au hasard** *(Missing At Random – MAR)* :

  Les données manquantes ne dépendent pas de la variable d'intérêt mais sont conditionnées par d'autres variables observées. On retrouve dans ces données manquantes les réponses d'une enquête qui ne peuvent être obtenues car la question est conditionnée à la question précédente.

- **Non manquantes au hasard** *(Not Missing At Random – NMAR)* :

  Les valeurs manquantes dépendent des valeurs elles-mêmes.

Gestion des données manquantes
(1/2)

Pour traiter les données manquantes, il est possible de ***supprimer*** ou d'***imputer*** les données.

La ***suppression*** de cas qui revient à omettre les valeurs manquantes dans l'analyse. Une telle approche peut entraîner des biais dans les résultats si l'échantillon supprimé ne constitue pas un sous-ensemble aléatoire de la population. D'autre part, la suppression entraîne une perte d'information qui peut se répercuter sur la variabilité des résultats. Little et Rubin (2019) recommandent de ne pas appliquer ce type de retraitement lorsqu'une série possède plus de 5% de valeurs manquantes.

L'***imputation*** permet de combler une valeur manquante, tout en cherchant à minimiser les biais. Cela permet d'exploiter des séries de données avec des valeurs manquantes parfois coûteuses, offrant ainsi la possibilité d'éviter la suppression de données. L'imputation simple revient à substituer les données manquantes par la moyenne, la médiane. Il existe aussi d'autres types d'imputations, comme par exemple l'imputation par régression ou l'imputation par maximisation attendue. Enfin, l'imputation multiple fournit plusieurs valeurs pour chaque donnée manquante.

Gestion des données manquantes
(2/2)

Lorsque l'imputation est réalisée, l'incertitude des données imputées peut être évaluée par la variance pour mesurer les effets que cela peut avoir sur l'analyse. D'autre part, les hypothèses faites et les résultats obtenus doivent faire l'objet de vérifications plus approfondies (propriétés statistiques, caractéristiques de distribution).

En bref, la gestion des valeurs manquantes c'est avoir :

1 - Des séries de données complètes ;
2 - Une mesure de fiabilité des valeurs imputées (et l'impact de cette dernière lorsque la variable est utilisée pour la création d'un indice composite ou d'une notation) ;
3 - Une discussion sur les valeurs aberrantes ;
4 - Une documentation pour expliquer les procédures de retraitement.

Imputation simples – Modélisation implicite

Réaliser une imputation revient à attribuer une valeur à une variable lorsque cette dernière est absente. Pour effectuer cette attribution, il convient d'utiliser une distribution prédictive des valeurs manquantes. Cette distribution doit être générée à partir des données observées, soit par modélisation implicite, soit par modélisation explicite.

Imputation simples – Modélisation implicite

La **modélisation implicite** s'appuie sur algorithme, avec des hypothèses sous-jacentes qui doivent être vérifiées pour s'assurer qu'elles soient adaptées au problème. Il existe différents types de modélisations implicites, notamment :

- **Hot deck imputation** : lorsqu'une donnée d'une entreprise, d'un individu, ou d'un pays par exemple est absente, on utilise une donnée d'une entreprise, d'un individu ou d'un pays *similaires*. Par exemple, si la donnée sur le revenu d'un ménage est absente, il est possible de tirer une valeur de la distribution des ménages présentant des caractéristiques similaires (sur l'âge, le sexe, le lieu de résidence, l'emploi...) ;

- **Substitution** : lorsque les données d'un individu ne sont pas disponibles, il est possible de substituer cet individu avec un autre individu qui n'était pas utilisé précédemment.

- **Cold deck imputation** : lorsque la donnée est manquante, utilisation d'une source externe. Lorsqu'il s'agit d'une enquête, il est possible d'utiliser une réalisation antérieure de cette même enquête.

Imputation simples – Modélisation explicite

La **modélisation explicite** repose sur des modèles statistiques dans lesquels les hypothèses sont formulées explicitement.

- **Unconditional mean/median/mode imputation** : la valeur manquante est remplacée par la moyenne, la médiane, ou le mode de l'échantillon des valeurs disponibles.

- **Linear interpolation** : si la valeur manquante est située entre deux points dont les données sont disponibles, dans une série temporelle par exemple, l'interpolation linéaire permet d'estimer cette valeur par une fonction continue (fonction affine).

- **Regression imputation** : les valeurs manquantes sont remplacées par les valeurs obtenues à partir d'une régression. La variable dépendante, ou variable à expliquer, va correspondre à la valeur manquante. Et les variables explicatives vont être des indicateurs individuels propres au pays, à l'individu, ou à l'entreprise, avec une forte relation avec la variable dépendante.

- **Expectation Maximisation imputation** : un processus itératif est utilisé pour remplacer les valeurs manquantes. Ce processus itératif consiste à estimer les valeurs des paramètres, remplacer les valeurs manquantes grâce aux estimations, puis réestimer les paramètres. Cette séquence est répétée jusqu'à ce que les résultats convergent.

Interpolation et extrapolation linéaires
(1/3)

L'interpolation linéaire est une méthode qui permet d'estimer la valeur d'un point en fonction de deux points déterminés. Cette fonction linéaire se calcule comme suit :

$$f(x) = f(a) + \big(f(b) - f(a)\big) \times \frac{(x-a)}{(b-a)}$$

Le package **Hmisc** sur le logiciel **R** permet de faire une interpolation linéaire avec la fonction **approx()**. Il est également possible d'utiliser la fonction **approxExtrap()** pour faire une extrapolation linéaire.

Interpolation et extrapolation linéaires
(2/3)

Exemple - Interpolation linéaire :

```
1
2   install.packages('Hmisc')
3   library(Hmisc)
4
5
6   # ------------------------------------
7   # ------- Interpolation lineaire -------
8   # ------------------------------------
9
10  # Valeurs initiales: annees et PIB de la France
11  date <- c(1999, 2001, 2002, 2003, 2005, 2007)
12  Value_PIB <- c(1408.2, 1544.6, 1594.3, 1637.4, 1772.0, 1945.7)
13
14  # Construction de la base initiale
15  database <- data.frame(date, Value_PIB)
16
17  # Dates souhaitees, ici toutes les annees de 1999    2007
18  date_complete <- seq(1999,2007,1)
19
20  # Affichage des donnees
21  plot (date, Value_PIB, type='o')
22  abline(v = date_complete, col = '#ff0000')
23
24
25  # Calcul par interpolation
26  resultat <- approx(date, Value_PIB, xout = date_complete, method='linear', ties='
        ordered')
27  # Recuperation des resultats
28  database_Finale <- data.frame(date_complete,resultat$y)
29
30
31  # Affichage du resultat
32  plot (date_complete, resultat$y, type='o')
33  abline(v = date_complete, col = '#ff0000')
```

Interpolation et extrapolation linéaires
(3/3)

Exemple - Extrapolation linéaire :

```
35  # ------------------------------------------
36  # ------- Extrapolation lineaire ---------
37  # ------------------------------------------
38
39  # Ajout des dates 1997 et 1998 pour l extrapolation
40  date_extra <- seq(1997,2007,1)
41
42  # extrapolation
43  resultat_extra <- approxExtrap(date, Value_PIB, xout = date_extra, method='linear',
        ties='ordered')
44  database_Finale_extra <- data.frame(date_extra,resultat$y)
45
46  # Affichage du resultat
47  plot (date_extra, resultat_extra$y, type='o')
48  abline(v = date_extra, col = "#ff0000")
```

Régression linéaire
(1/4)

La méthode des moindres carrés ordinaires (*Ordinary Least Squares - OLS*) permet de faire une régression linéaire. Cette approche permet d'estimer la relation d'une variable dépendante, notée *Y*, avec des variables explicatives, notées *X*.

La formule suivante présente la relation entre la variable dépendante et les variables explicatives :

$$Y_i = X_i'\beta$$

β est un vecteur de coefficients. Ces coefficients peuvent être calculés avec la méthodes des moindres carrés ordinaires. L'estimation se fait comme suit :

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Régression linéaire
(2/4)

Preuve :

$$\sum_{i=1}^{n} \varepsilon_{i0}^2 = \sum_{i=1}^{n} (y_i - x_i' \beta_0)^2$$

$$Min_{b_0}\, S(\beta_0) = \varepsilon_0' \varepsilon_0 = (Y - X b_0)'(Y - X\beta_0)$$

$$S(\beta_0) = Y'Y - 2Y'X\beta_0 + \beta_0' X'X b_0$$

$$\frac{\delta S(\beta_0)}{\delta \beta_0} = -2X'Y + 2X'X\beta_0 = 0$$

$$\rightarrow \beta = (X'X)^{-1} X'Y$$

Régression linéaire
(3/4)

Lorsqu'une valeur de la série *Y* est manquante, il est possible d'utiliser la régression linéaire pour l'estimer et la remplacer par la valeur obtenue. Il convient alors de déterminer des variables permettant de la calculer.

Dans l'exemple suivant, nous utilisons la fonction lm() qui permet de réaliser une estimation avec la méthode du maximum de vraisemblance.
Les résultats sont proches de ceux obtenus avec une régression linéaire.

Régression linéaire
(4/4)

```r
# recuperation des donnees
database <- read.csv(file = "C:/Users/data_poids.csv", header = TRUE, sep = ";")

# Affichage des premieres donnees pour chaque variables
head(database)

# statistiques descriptives
summary(database)

# recuperation des donnees
Y <- as.vector(database$poids) # variable dependante
X1 <- as.vector(database$taille) # variable independante - taille
X2 <- as.vector(database$Sexe) # variable independante - Sexe
X3 <- as.vector(database$Age) # variable independante - Age
X <- as.matrix(cbind(X1, X2, X3)) # matrice des variables explicatives

# regression sans constante
results <- lm(Y ~ X -1)
summary(results)

# regression avec constante
results <- lm(Y ~ X)
summary(results)
```

Exercices

Exercice 1 :

Développez une fonction d'interpolation linéaire qui prend en paramètres x, deux points x_a et x_b, ainsi que leur deux valeurs associées y_a et y_b. La fonction renverra y, la valeur associée à x, et sera calculée à partir de la fonction affine dépendant des couples $(x_\mathrm{a}, \mathrm{y_a})$ et (x_b, y_b).

Exercice 2 :

Développez une fonction qui permet d'estimer les coefficients β d'une régression linéaire, en passant par la formule : $\beta = (X'X)^{-1}X'Y$

Exercice 3 :

Développez deux fonctions permettant de calculer la moyenne, l'écart-type. Chacune de ces fonctions aura comme input un vecteur de données.
Pour rappel :

$$\mathrm{moyenne} = \mu = \frac{1}{\mathrm{N}}\sum_{i=1}^{N} x_i$$

$$ecart - type = \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Chapitre 4

La normalisation des données

Introduction

Lorsque les vecteurs de données ont des échelles différentes, il est difficile de les comparer. Et dans la création d'un indice, il peut être nécessaire d'avoir recours à la normalisation des données pour réaliser cette mise à l'échelle et rendre les données comparables. D'autre part, cela évite d'avoir des vecteurs de données surpondérés par construction.

Il existe plusieurs méthodes de normalisation qui ne conduisent pas toutes au même résultat. Il est donc important de déterminer la méthode adéquate, et de vérifier la robustesse des indices en testant plusieurs méthodes de normalisation.

L'étape de normalisation doit permettre de :

- Choisir les procédures de normalisation appropriées (respect du cadre théorique et des propriétés des données) ;
- Discuter de la présence de valeurs aberrantes ;
- Faire des ajustements d'échelle.

Sommaire

La normalisation des données

01	CLASSEMENT	67
02	STANDARDISATION	68
03	MIN-MAX	69
04	DISTANCE TO REFERENCE	70
05	ECHELLE CATÉGORIELLE	71
06	SUP-INF	72
07	EXERCICES	73

Classement (Ranking)

La méthode du **classement** (ranking) est moins concernée par les valeurs extrêmes.

$$X_{rank} = rank(X)$$

Le ranking peut se faire sur une série de données propres à un pays ou individu (exemple 1), mais aussi sur chaque période pour un panel de pays ou d'individus (exemple 2).

Exemple 1

Time	X	Rank
1	0.76	9
2	0.57	8
3	0.34	4
4	0.55	6
5	0.08	3
6	0.46	5
7	0.03	1
8	0.57	7
9	0.06	2

Exemple 2

Time	X1	X2	X3
1	0.21	0.44	0.22
2	0.69	0.35	0.20
3	0.88	0.57	0.55
4	0.13	0.45	0.32
5	0.97	0.40	0.12
6	0.28	0.00	0.45
7	0.44	0.19	0.13
8	0.90	0.45	0.56
9	0.91	0.22	0.94

Ranking

X1	X2	X3
1	3	2
3	2	1
3	2	1
1	3	2
3	2	1
2	1	3
3	2	1
3	1	2
2	1	3

Standardisation (Z-Score)

La **standardisation** (également appelée Z-Score), est une approche qui permet d'obtenir un nouvel indicateur avec une moyenne égale à 0 et un écart-type à 1.

$$X_{standard} = \frac{X_i - \bar{X}}{\sigma_X}$$

Avec :

- La moyenne : $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$
- L'écart-type : $\sigma_X = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{X})^2}$

Min-Max

La méthode **min-max** est un processus qui permet d'avoir des données comprises entre 0 et 1, et de conserver la distance entre les valeurs :

$$X_{norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

Mais les résultats peuvent être très sensibles aux valeurs extrêmes et aux valeurs aberrantes.

Distance to reference

La distance par rapport à une référence permet d'attribuer à des données une valeur calculée selon la distance avec un point prédéterminé. Ce point peut-être la valeur d'un autre pays, un objectif réglementaire, ou la moyenne de l'échantillon :

$$X_{Distance} = \frac{X_{t,i}}{X_{t0,j}}$$

Cet indicateur permet de regarder l'évolution par rapport à un pays, et conditionnellement à la valeur initiale ciblée.

Echelle catégorielle

L'**échelle catégorielle** (Categorical scale) attribue un score pour chaque indicateur. Le résultat peut-être numérique (valeur allant de 1 à 5 par exemple) ou qualitatif (objectif atteint vs objectif non atteint).
Cette méthode s'appuie généralement sur les percentiles de la distributions, et masque ainsi une partie de la variance des indicateurs.

Par exemple :

$$\begin{cases} 0 & if \ \ X_t < P^{15} \\ 20 & if \ \ P^{15} \leq X_t < P^{25} \\ 40 & if \ \ P^{25} \leq X_t < P^{50} \\ 60 & if \ \ P^{50} \leq X_t < P^{75} \\ 80 & if \ \ P^{75} \leq X_t < P^{85} \\ 100 & if \ \ P^{85} \leq X_t \end{cases}$$

Sup-Inf

Les indicateurs autour de la moyenne sont transformés en prenant la valeur 0, tandis que les valeurs supérieurs ou inférieures à un seuil reçoivent 1 ou -1 respectivement.
Cette méthode n'est pas sensible aux valeurs extrêmes, mais les résultats peuvent être affectés par le choix du seuil.

$$\begin{cases} 1 & if\ w \geq (1+p) \\ 0 & if\ (1-p) \leq w < (1+p) \\ -1 & if\ w < (1-p) \end{cases}$$

Avec $w = \frac{X_t}{X_{t0}}$ et X_{t0} la valeur moyenne de plusieurs pays, ou plusieurs individus, pour une année fixée et pour l'indicateur sélectionné.

La valeur p, correspondant au seuil, permet d'avoir une zone neutre autour de la moyenne, qui conduit à l'obtention de la valeur 0 pour l'indicateur lorsqu'il se situe dans l'intervalle.

Exercices

Exercice 1 :

Développer une fonction qui permette de transformer un vecteur de données en un vecteur de variables standardisées.

Aide : *Standardisation* $\rightarrow X_{standard} = \frac{X_i - \bar{X}}{\sigma_X}$

Exercice 2 :

Développer une fonction qui permette de transformer un vecteur de données en un vecteur de variables avec la transformation min-max.

Aide : *Min-Max* $\rightarrow X_{norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$

Exercices

Exercice 3 :

Développer une fonction d'échelle catégorielle qui prend en paramètres un vecteur de données, les 5 quantiles, et qui renvoie les valeurs de 0/20/40/60/80/100 conditionnellement à la valeur de chaque de données et de sa position par rapport aux quantiles.

Exercice 4 :

Développer une fonction Sup-Inf qui prend en paramètre un vecteur et un seuil, et qui retourne les valeurs -1, 0 et 1 du vecteur conditionnellement à la règle :

$$\begin{cases} 1 & if\ w < (1+p) \\ 0 & if\ (1-p) \leq w \leq (1+p) \\ -1 & if\ w < (1-p) \end{cases} \text{ avec } w = \frac{X_t}{\mu}$$

Chapitre 5

L’analyse multivariée et l’ACP

Sommaire

L’analyse multivariée et l’ACP

01	CONCEPTS ET DÉFINITIONS	77
02	LA CORRÉLATION SUR R	82
03	EXERCICES	83
04	L’ACP SUR R	84
05	ANALYSE DES RÉSULTATS DE L’ACP	90

Concept et Définitions
(1/5)

Analyse en Composante Principale – ACP (Principal Component Analysis – PCA)

L'analyse en composantes principales permet de résumer et de visualiser les informations d'un ensemble de données contenant des individus/observations décrits par de multiples variables quantitatives corrélées.

Chaque variable peut être considérée comme une dimension différente. L'ACP est utilisée pour extraire les informations importantes de séries de données (multivariées) et pour exprimer ces informations sous la forme d'un ensemble de quelques nouvelles variables appelées composantes principales. Ces nouvelles variables correspondent à une combinaison linéaire des variables d'origine.

Une fois l'ACP réalisée, le nombre de composantes principales est inférieur ou égal au nombre de variables originales.

Concept et Définitions
(2/5)

Analyse en Composante Principale – ACP (Principal Component Analysis – PCA)

Supposons un nombre de séries quantitatives, avec certaines séries corrélées entre elles. Dans une telle situation, il est possible d'utiliser l'ACP pour réduire le nombre de dimensions au nombre de dimension minimum possible pour conserver l'information, et obtenir alors des nouveaux indicateurs ou nouvelles séries non corrélées.

Avec l'ACP, on projette l'ensemble des données sur un hyperplan, et on récupère les nouveaux vecteurs directeurs qui correspondent à ces variables principales (réduction de dimensions).

L'ACP permet une meilleure visibilité de nos données, avec le problème de corrélation résolu.

Concept et Définitions
(3/5)

Covariance

Elle permet de déterminer les écarts de deux variables par rapport à leur espérance respective. Elle correspond aux variations simultanées de ces deux variables, et se calcule comme suit :

$$cov_{X,Y} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Concept et Définitions
(4/5)

Corrélation

Elle mesure la relation entre deux variables, et est comprise entre -1 et 1. Deux variables fortement corrélées ont une corrélation proche de 1 en valeur absolue, et proche de 0 lorsqu'elle sont décorrélées.

Attention, la corrélation est un lien statistique, et est indépendante de la causalité. Deux variables peuvent être fortement corrélées sans qu'aucun lien n'existe entre elles.

De nombreux exemples de « *spurious correlations* » existent :
https://www.tylervigen.com/spurious-correlations

Concept et Définitions
(5/5)

Corrélation

La corrélation de Pearson calcule la relation linéaire entre deux variables continues, tandis que la corrélation de Spearman s'appuie sur le rang de chacune des deux variables.

La corrélation de Pearson se calcule comme suit :

$$corrélation\left(X,Y\right) \rightarrow \rho_{X,Y} = \frac{cov_{X,Y}}{\sigma_X \sigma_Y}$$

La corrélation de Spearman se calcule comme suit :

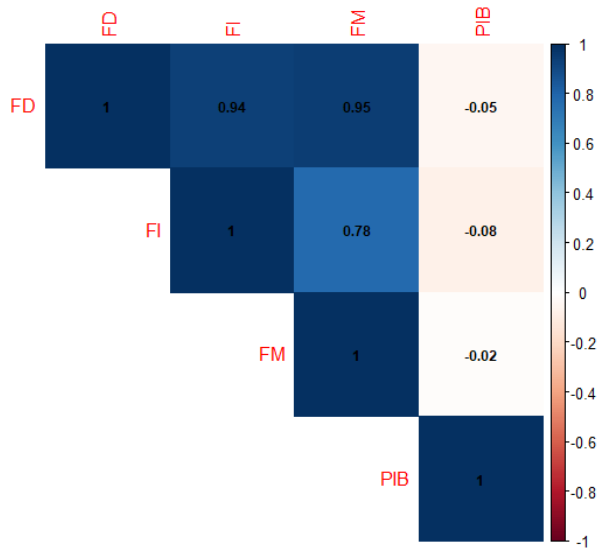
$$corrélation\left(X,Y\right) \rightarrow \rho_{X,Y} = \frac{cov_{rg(X),rg(Y)}}{\sigma_{rg(X)} \sigma_{rg(Y)}}$$

La corrélation sur R

Pour calculer la **corrélation** sur R, il est possible d'utiliser la fonction **cor()**. Cette fonction peut prendre en paramètre une matrice de n variables, et renverra la matrices des corrélations pour chaque couple de variables.

L'argument *method* peut permettre de choisir le type de correlation que l'on souhaite calculer:

method = c("pearson", "kendall", "spearman")



Le package **corrplot** permet de visualiser la matrice de corrélation.

Exercices

Exercice 1 :

Coder la fonction permettant de calculer la covariance entre deux variables. La fonction doit prendre en input deux vecteurs de données.

Aide : $cov_{X,Y} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$

Exercice 2 :

Coder la fonction permettant de calculer la corrélation (de Pearson) entre deux variables. La fonction doit prendre en input deux vecteurs de données.

Aide : $corrélation\ (X,Y) \rightarrow \rho_{X,Y} = \frac{cov_{X,Y}}{\sigma_X \sigma_Y}$

L'ACP sur R
(1/6)

Pour l'analyse en composante principale, nous allons utiliser dans ce cours les packages FactoMineR et factoextra.

Le package FactoMineR permet d'appliquer les différentes méthodes nécessaire à l'ACP, et le package factoextra permet d'extraire, de visualiser et d'interpréter les résultats obtenus avec l'ACP.

Les exemples ci-dessous s'appuient sur les données contenues dans le package factoextra et correspondent aux performances de 27 athlètes sur deux événements sportifs.

```
1
2
3   install.packages("FactoMineR")
4   library(FactoMineR)
5
6   install.packages("factoextra")
7   library(factoextra)
8
9
10  # donnees obtenues a partir du package factoextra
11  data(decathlon2)
12  head(decathlon2)
13
14  # Variables quantiatives pour PCA
15  database_decat <- decathlon2[1:23, 1:10]
16  head(database_decat)
```

L’ACP sur R
(2/6)

Pour réaliser une ACP, il faut utiliser la fonction PCA() du package FactoMineR.

La fonction PCA prendre les paramètres suivants :

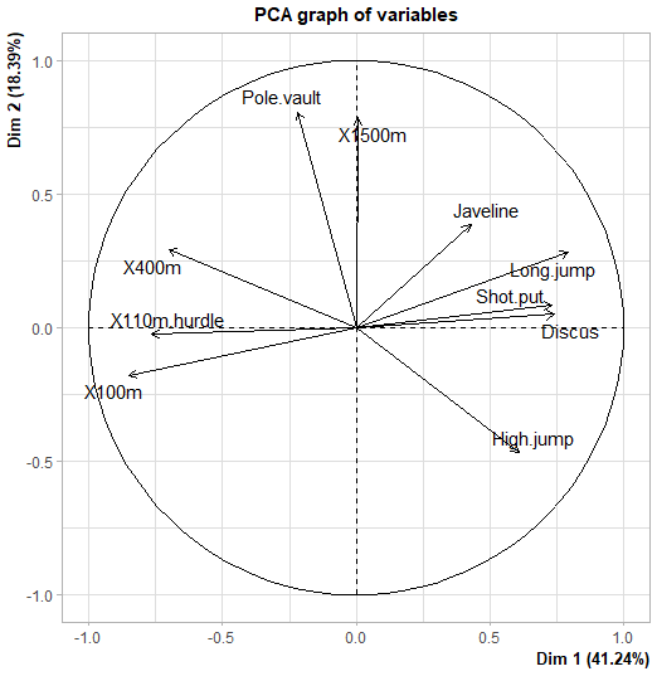
PCA(X, scale.unit, ncp, graph)

- **X** : données pour l’ACP. Les lignes sont des individus et les colonnes des variables numériques ;
- **scale.unit** : variable boolean permettant de mettre à l'échelle les données avec un z-score ;
- **comparables.ncp** : nombre de dimensions que l’on souhaite conserver ;
- **finaux.graph** : variable boolean pour afficher un graphique.

L'ACP sur R
(3/6)

```r
# Exemple d utilisation d'ACP
results_PCA <- PCA(database_decat, scale.unit = TRUE, graph = TRUE)

# Affichage des resultats
print(results_PCA)
```

```
> # Exemple d'utilisation d'ACP
> results_PCA <- PCA(database_decat, scale.unit = TRUE, ncp = 3, graph = TRUE)
> print(results_PCA)
**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 23 individuals, described by 10 variables
*The results are available in the following objects:

   name                 description
1  "$eig"               "eigenvalues"
2  "$var"               "results for the variables"
3  "$var$coord"         "coord. for the variables"
4  "$var$cor"           "correlations variables - dimensions"
5  "$var$cos2"          "cos2 for the variables"
6  "$var$contrib"       "contributions of the variables"
7  "$ind"               "results for the individuals"
8  "$ind$coord"         "coord. for the individuals"
9  "$ind$cos2"          "cos2 for the individuals"
10 "$ind$contrib"       "contributions of the individuals"
11 "$call"              "summary statistics"
12 "$call$centre"       "mean of the variables"
13 "$call$ecart.type"   "standard error of the variables"
14 "$call$row.w"        "weights for the individuals"
15 "$call$col.w"        "weights for the variables"
>
```



L’ACP sur R
(4/6)

Valeurs propres – eigenvalues (→ Results_PCA$eig)
Les valeurs propres (eigenvalues) mesurent la quantité de variations de chaque composante principale. Ces valeurs doivent permettre de déterminer le nombre de composantes principales à prendre en compte. La première composante sera celle qui apporte le plus de variations et donc d’informations. Ensuite, chaque composante aura un apport de variation décroissant.

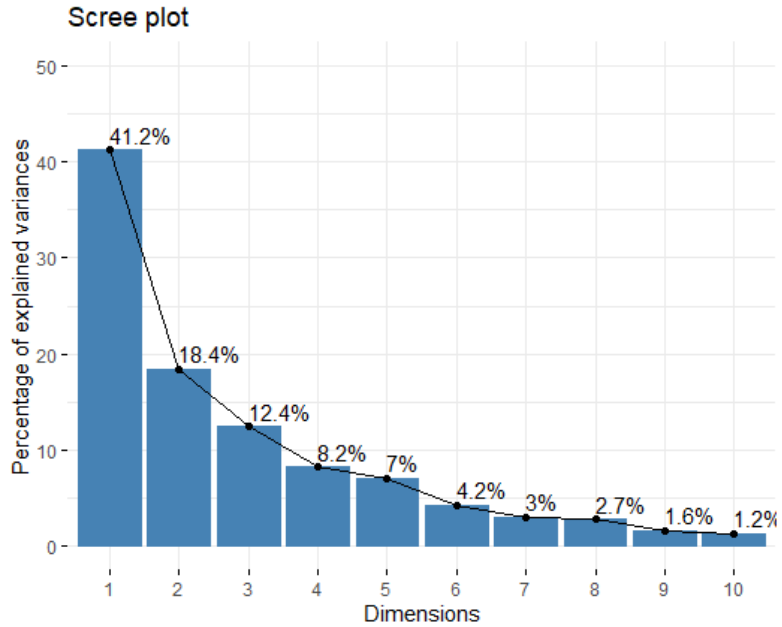
Pour déterminer les valeurs propres retenues et l’information captée par ces composantes principales, il est possible d’utiliser la fonction **get_eigenvalue()**. Cette fonction donne la proportion d’explication par chaque variable. Lorsqu’il y a n variables, la somme de la première colonne est égale à n. La deuxième colonne donne le pourcentage de variabilité total auquel contribue chaque dimension sur la variabilité totale.

La fonction **fviz_eig()** peut également être utilisée pour voir graphiquement l’apport marginal de chaque variable sur la variabilité globale des résultats.

L'ACP sur R
(5/6)

```
1
2  # Exemple d'utilisation d'ACP
3  results_PCA <- PCA(database_decat, scale.unit = TRUE, graph = TRUE)
4
5  # Affichage des resultats
6  print(results_PCA)
7
8  # affichages des valeurs propres
9  eig.val <- get_eigenvalue(results_PCA)
10 print(eig.val)
11
12 # representaiton graphique
13 fviz_eig(results_PCA, addlabels = TRUE, ylim = c(0, 50))
```

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	4.1242133	41.242133	41.24213
Dim.2	1.8385309	18.385309	59.62744
Dim.3	1.2391403	12.391403	72.01885
Dim.4	0.8194402	8.194402	80.21325
Dim.5	0.7015528	7.015528	87.22878
Dim.6	0.4228828	4.228828	91.45760
Dim.7	0.3025817	3.025817	94.48342
Dim.8	0.2744700	2.744700	97.22812
Dim.9	0.1552169	1.552169	98.78029
Dim.10	0.1219710	1.219710	100.00000



L’ACP sur R
(6/6)

Les valeurs propres peuvent servir à déterminer le nombre de composantes principales retenues. Plusieurs méthodes peuvent être envisagées :

- Valeur propre supérieure à 1 → apporte plus d’information que ce qu’apportait la variable initiale ;
- Variance totale supérieure à 70% → l’information captée par les variables est suffisante pour conserver l’information d’origine ;
- Approche graphique → nombre de composantes qui apporte plus d’informations que celles ayant une petite contribution, et de taille comparable à d’autres.

Analyse des résultats de l'ACP
(1/5)

results_PCAvarcor → donne les corrélations entre les variables et les composantes

results_PCAvarcos2 → représente la qualité de la représentation des variables sur la carte factorielle. Il est calculé comme le carré des coordonnées :

$$var.cos2 = var.cor \times var.cor$$

results_PCA$ var$contrib → contient les contributions (en pourcentage) des variables aux composantes principales. La contribution d'une variable (var) à une composante principale donnée est (en pourcentage) :

$$var.contrib = \frac{var.cos2 \times 100}{total(cos2\ of\ the\ component)}$$

Analyse des résultats de l'ACP
(2/5)

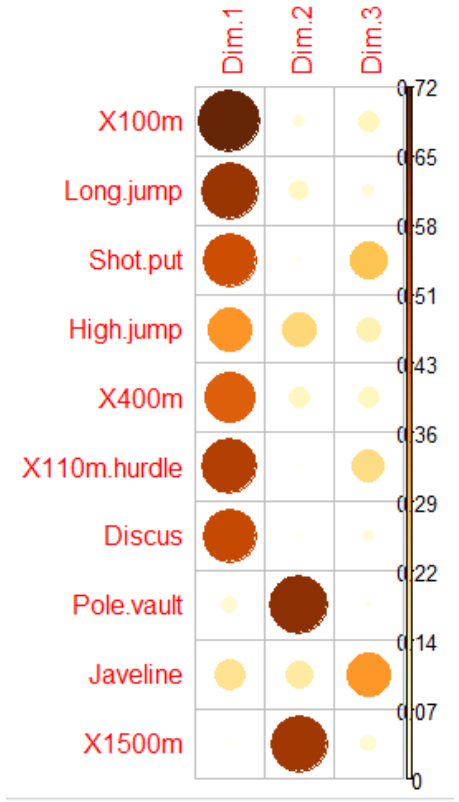
La fonction var_results_PCA() permet d'obtenir directement ces résultats.

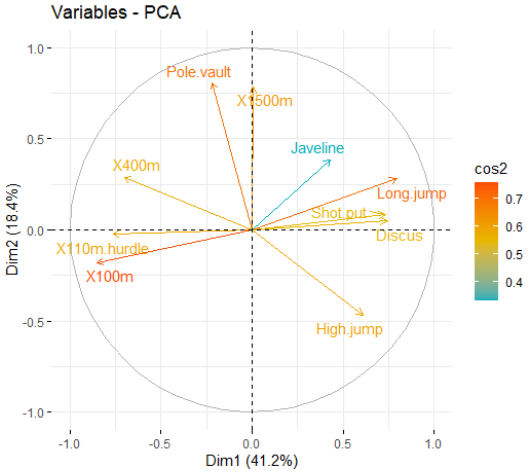
```
# Recuperation de tous les resultats associes a la sortie var
var_results_PCA <- get_pca_var(results_PCA)

# correlation entre les resultats et les composantes
var_results_PCA$cor

# qualite de la representation des variables dans chaque dimension
var_results_PCA$cos2

# contribution de chaque variable a chaque dimension
var_results_PCA$contrib

# calcul de la contribution
sum(var_results_PCA$cos2[,1])
```

Analyse des résultats de l'ACP
(3/5)

On peut visualiser ensuite l'apport de chaque variable à chaque dimension en utilisant la fonction **corrplot()** vue précédemment.

```
1
2  # contribution des variables
3  corrplot(results_PCA$var$cos2, is.corr=FALSE)
4
5  # Coloration en fonction des valeurs de cos2
6  fviz_pca_var(results_PCA, col.var = "cos2",
7               gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
8               repel = TRUE # Avoid text overlapping
9  )
```

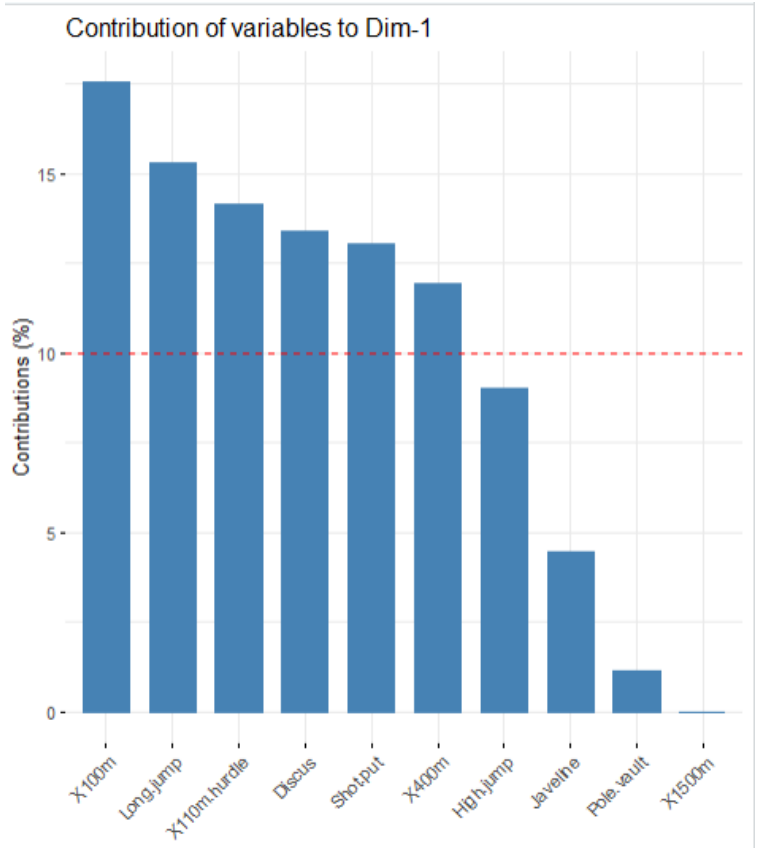




Analyse des résultats de l'ACP
(4/5)

La fonction **fviz_contrib()** permet également de visualiser la contribution de chaque variable aux différentes dimensions.

```
# Contributions des variables a la composante 1
fviz_contrib(results_PCA, choice = "var", axes = 1, top = 10)

# Contributions des variables a la composante 2
fviz_contrib(results_PCA, choice = "var", axes = 2, top = 10)

# Contributions des variables a la composante 3
fviz_contrib(results_PCA, choice = "var", axes = 3, top = 10)

# Contributions des variables a la composante 4
fviz_contrib(results_PCA, choice = "var", axes = 4, top = 10)

# Contributions des variables a la composante 5
fviz_contrib(results_PCA, choice = "var", axes = 5, top = 10)
```



Analyse des résultats de l'ACP
(5/5)

Enfin, la fonction **dimdesc()** permet d'afficher la corrélation et leur significativité entre les variables et les différentes composantes.

```
1  res.desc <- dimdesc(results_PCA, axes = c(1:5), proba = 0.05)
2
3  # Affichage des correlations de variables avec la troisieme composante
4  res.desc$Dim.3
```

```
> res.desc$Dim.3
$quanti
              correlation     p.value
Javeline        0.6041243 0.002266783
Shot.put        0.5175978 0.011419547
X110m.hurdle    0.4488873 0.031667860

attr(,"class")
[1] "condes" "list"
```

La fonction **write.infile()** permet stocker les résultats obtenus avec l'ACP.

Chapitre 6

Pondération et Agrégation

Sommaire

Pondération et Agrégation

01 INTRODUCTION 97

02 LA PONDÉRATION 98

03 EXERCICES 102

04 PONDÉRATION AVEC L'ACP 103

ETAPE 1 105

ETAPE 2 106

ETAPE 3 107

05 MÉTHODE D'AGRÉGATION 110

06 EXERCICES 113

Introduction

Les méthodes de pondération et d'agrégation doivent être utilisées en respectant le cadre théorique initialement posé. Lors de la création d'un indicateur composite, ces méthodes permettent de :

Sélectionner une méthode de pondération et d'agrégation qui tienne compte des propriétés des données ;
Evaluer la corrélation des données et permettre de corriger et/ou discuter des impacts que cela peut avoir sur les résultats finaux

La pondération
(1/4)

La pondération des indicateurs peut avoir un impact significatif sur la valeur de l'indice composite. Dés lors, la comparaison des pays, des entreprises ou des individus peut être faussée ou biaisée.

La pondération peut être basée uniquement sur des méthodes statistiques (comme l'Analyse en Composante Principale), tandis que d'autres méthodes peuvent s'appuyer sur le jugement d'experts. L'avis d'experts aura pour conséquence une surpondération ou une sous-pondération (récompenser ou pénaliser) certaines variables pour mieux refléter les facteurs théoriques ou les objectifs fixés par les priorités (politiques, innovation, environnementales, etc…).

La pondération
(2/4)

De nombreux indicateurs composites reposent sur une pondération « Equal Weighting » (EW). Cela consiste à avoir un indicateur composite construit à partir de n variables, où chaque variable aura un poids de $\frac{1}{n}$. Cette méthode ne signifie pas qu'il n'y a « pas de poids », mais plutôt que toutes les variables ont implicitement le même poids.

Pour permettre d'avoir un poids similaire dans l'indicateur, il convient de s'assurer que les indicateurs aient la même échelle. Les méthodes de normalisation doivent être homogènes pour que les différentes unités des variables ne créent pas de biais. D'autre part, si les variables sont agrégées en dimensions, et que ces dimensions sont ensuite agrégées dans l'indice avec la méthode EW, les variables ne contribueront pas avec le même poids dans l'indice composite.

Les pondérations peuvent également être utilisées avec une approche qui favorise la qualité des données, en attribuant par exemple des poids plus élevés aux données fiables et avec une meilleure couverture. Mais cette méthode pénalisera par construction les informations difficiles à identifier et à mesurer.

Pondération
(3/4)

Enfin, lorsqu'on utilise des poids égaux, les variables ayant un degré élevé de corrélation vont introduire un double comptage dans l'indice. Les indicateurs colinéaires seront inclus avec un poids plus élevé.

Avant de déterminer un poids pour chaque indicateur, l'évaluation de la corrélation (de Pearson ou de Spearman) peut permettre d'éviter ce double-comptage. Une règle empirique peut être introduite pour définir un seuil au-delà duquel la corrélation est considérée comme à l'origine d'un double comptage. Il est également possible d'ajuster les corrélations en attribuant un poids inférieur au indicateurs corrélés, ou de réduire le nombre de variables en privilégiant les indicateurs avec une faible corrélation.

Pondération
(4/4)

Attention, si les pondérations reflètent la contribution de chaque indicateur à l'indice composite, le double comptage ne doit pas uniquement dépendre d'une analyse statistique. Il convient également de se rapporter au cadre théorique dans lequel on se situe, et au phénomène que cet indice vise à capturer.

Par exemple, un indicateur qui cherche à mesurer des aspects différents d'innovation et d'adoption des technologies d'innovation s'appuie plusieurs variables, notamment : « Pourcentage d'entreprises utilisant Internet » et « Pourcentage d'entreprises disposant d'un site web ». Or, ces deux variables affichent une corrélation de 0.88 en 2003. Faut-il considérer que ces indicateurs mesurent des aspects différents ou faut-il modifier les poids dans la construction de l'indice composite ?

Exercices

Exercice 1 :

Faire une fonction qui prenne en argument une matrice de n variables, et qui renvoie un indicateur qui agrège chaque variable avec un poids égal.

Exercice 2 :

Faire un indicateur composite avec 5 variables aléatoires, en utilisant la fonction `rnorm()`, la fonction `set.seed()` et la fonction de l'exercice 1.
Utiliser les mêmes variables simulées, et faire deux sous-indicateurs composites différents : le premier avec les 3 premières variables, et le second avec les 2 variables suivantes. Agrégée ensuite ces deux sous-indicateurs.
Comparer les résultats. Pourquoi les résultats sont-ils différents ? Quelle variable est surpondérée ou sous-pondérée dans l'indice 2 ?

Pondération avec l'ACP
(1/2)

Concernant l'agrégation des variables, il est possible d'utiliser des modèles statistiques tels que l'analyse en composantes principales. Cette méthode permet de regrouper des indicateurs individuels en fonction de leur degré de corrélation (méthode d'agrégation linéaire).

L'objectif est de conserver la plus grande variation possible dans l'ensemble des indicateurs en utilisant le plus petit nombre de facteurs.

Pondération avec l'ACP
(2/2)

Les étapes de la pondération basée sur l'ACP se fait en s'appuyant sur les étapes suivantes:

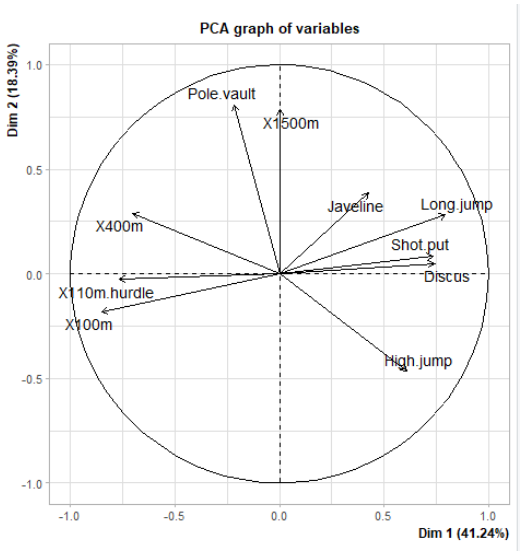
- **Etape 1** : vérifier la structure des corrélations des données. Si la corrélation entre deux indicateurs est faible, il est peu probable qu'ils partagent des facteurs communs.

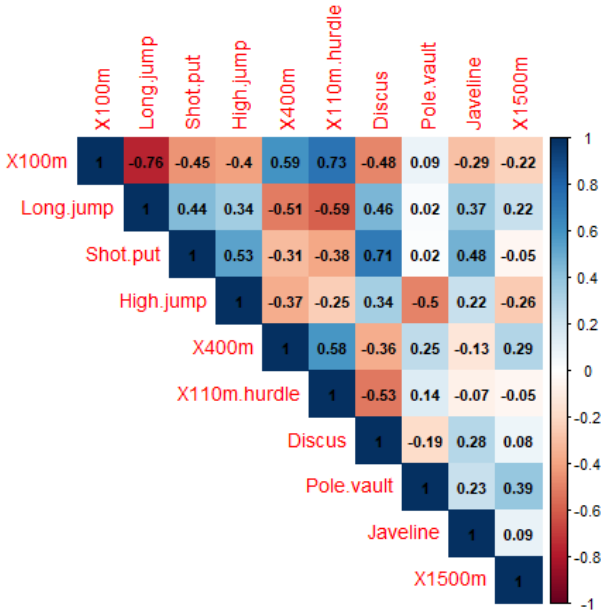
- **Etape 2** : identifier les facteurs (moins nombreux que les variables) qui représentent les données initiales. Le sous-ensemble de composantes principales retenues représentent la plus grande variance. Les règles suivantes permettent de sélectionner les facteurs :
  - Valeur propre supérieure à 1
  - Contribution à la variance globale supérieure à 10%
  - Contribution cumulative supérieure à 70% de la variance globale

- **Etape 3** : rotation des facteurs pour modifier la pondération des facteurs et faciliter l'interprétation de chaque composante.

Pondération avec l'ACP – Etape 1

```r
# données obtenues à partir du package factoextra
data(decathlon2)
head(decathlon2)

# Variables quantiatives pour PCA
database_decat <- decathlon2[1:23, 1:10]
head(database_decat)


# Représentation graphique de la matrice, avec le package corrplot
corrplot(cor(database_decat), method="color",
         type="upper",
         addCoef.col = "black", number.cex=0.75)

# ----------------------------------------------------
# -                        PCA                       -
# ----------------------------------------------------

# Exemple d'utilisation d'ACP
results_PCA <- PCA(database_decat, scale.unit = TRUE, graph = TRUE)

# Affichage des résultats
print(results_PCA)

# Affichages des valeurs propres
eigenvalue <- get_eigenvalue(results_PCA)
print(eigenvalue)
```



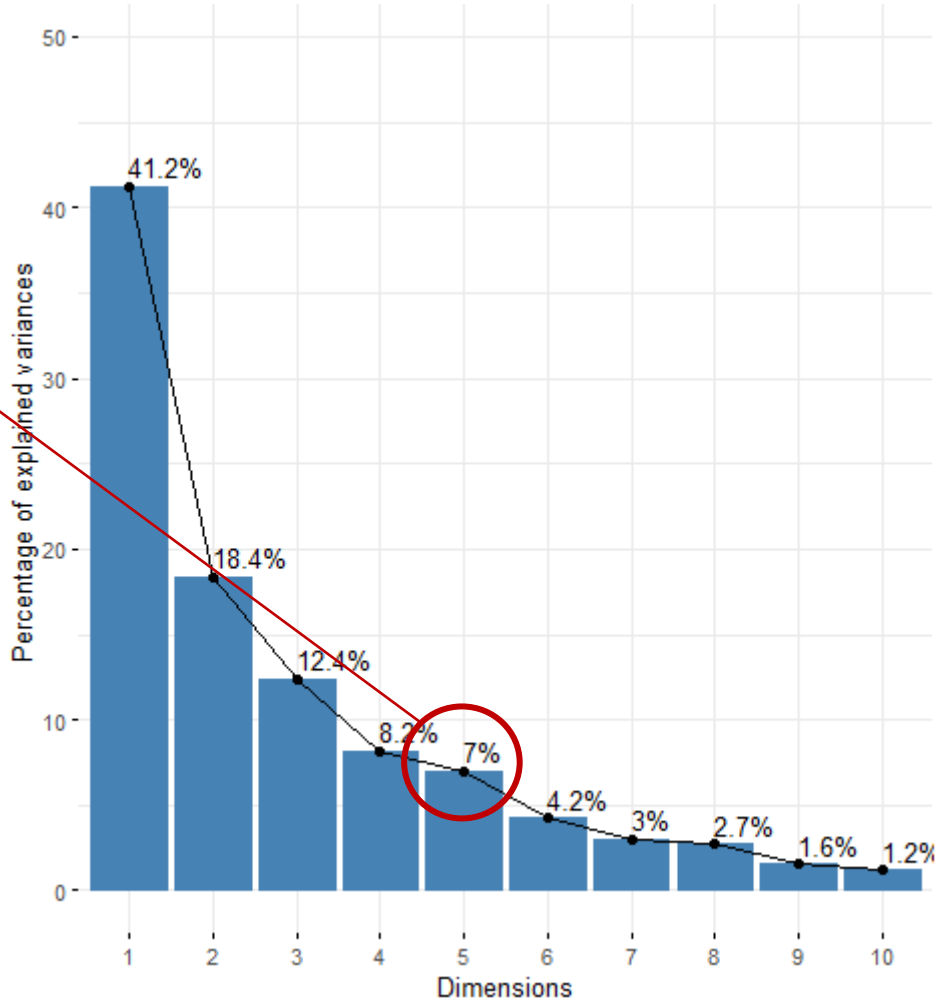


Pondération avec l'ACP – Etape 2

Méthode 1 : Les trois premières dimensions respectent bien les règles.

Méthode 2 : Approche graphique, avec une contribution marginale décroissante.

```
> print(eigenvalue)
       eigenvalue variance.percent cumulative.variance.percent
Dim.1   4.1242133        41.242133                    41.24213
Dim.2   1.8385309        18.385309                    59.62744
Dim.3   1.2391403        12.391403                    72.01885
Dim.4   0.8194402         8.194402                    80.21325
Dim.5   0.7015528         7.015528                    87.22878
Dim.6   0.4228828         4.228828                    91.45760
Dim.7   0.3025817         3.025817                    94.48342
Dim.8   0.2744700         2.744700                    97.22812
Dim.9   0.1552169         1.552169                    98.78029
Dim.10  0.1219710         1.219710                   100.00000
```



Pondération avec l'ACP – Etape 3
(1/3)

En fonction de la méthode choisie, il est possible de n'afficher que les dimensions qui nous intéresse dans les résultats en ajoutant le paramètre **ncp** dans la fonction PCA()

→ Méthode 1 : **results_PCA <- PCA(database_decat, scale.unit = TRUE, ncp=3, graph = TRUE)**

→ Méthode 2 : **results_PCA <- PCA(database_decat, scale.unit = TRUE, ncp=3, graph = TRUE)**

```
# Récupération de tous les résultats associés à "var"
var_results_PCA <- get_pca_var(results_PCA) # résultats associés à var pour la PCA

# Corrélation entre les résultats et les composantes
var_results_PCA$cor

# Qualité de la représentation des variables dans chaque dimension
var_results_PCA$cos2

# Contribution de chaque variable à chaque dimension
var_results_PCA$contrib
```

Pondération avec l'ACP – Etape 3
(2/3)

La qualité de la représentation des variables dans chaque dimension permet de pondérer chaque dimension dans l'indicateur global.

Variabilité dans chaque dimension de la variable i :
$var.cos2_i = var.cor_i \times var.cor_i$

Variabilité de chaque dimension à partir des variables :
$Var\ Dim_j = \sum_{i=1, i\in j}^{n} var.cos2_i$

Poids de chaque dimension :
$W_j = Var \frac{Dim_j}{\sum_{j=1}^{m} Var\ Dim_j}$

```
# Qualité de la représentation des variables dans chaque dimension
var_results_PCA$cos2
```

	Dim.1	Dim.2	Dim.3
X100m	0.72	0.03	0.09
Long.jump	0.63	0.08	0.04
Shot.put	0.54	0.01	0.27
High.jump	0.37	0.22	0.11
X400m	0.49	0.08	0.08
X110m.hurdle	0.58	0.00	0.20
Discus	0.55	0.00	0.03
Pole.vault	0.05	0.65	0.01
Javeline	0.18	0.15	0.36
X1500m	0.00	0.62	0.05
	4.12	**1.84**	**1.24**
	57.27%	**25.53%**	**17.21%**

Pondération avec l'ACP – Etape 3
(3/3)

Rotation des facteurs pour simplifier l'interprétation : on détermine les coefficients de chaque facteurs pour faire la combinaison linéaire permettant de calculer chaque composante finale.

- **Première étape** : $W_{ij} = \frac{var.cos2_i^2}{Var\ Dim_j}$ où W_{ij} est le poids de chaque variable i dans la composante j
- **Deuxième étape** : La variable est retenue dans une dimension/composante si sa contribution est supérieure à ses autres contributions.

	Dim.1	Dim.2	Dim.3
X100m	0.72	0.03	0.09
Long.jump	0.63	0.08	0.04
Shot.put	0.54	0.01	0.27
High.jump	0.37	0.22	0.11
X400m	0.49	0.08	0.08
X110m.hurdle	0.58	0.00	0.20
Discus	0.55	0.00	0.03
Pole.vault	0.05	0.65	0.01
Javeline	0.18	0.15	0.36
X1500m	0.00	0.62	0.05
	4.12	**1.84**	**1.24**
	57.27%	**25.53%**	**17.21%**

	Dim.1	Dim.2	Dim.3
X100m	**0.13**	0.00	0.01
Long.jump	**0.10**	0.00	0.00
Shot.put	**0.07**	0.00	0.06
High.jump	**0.03**	0.03	0.01
X400m	**0.06**	0.00	0.01
X110m.hurdle	**0.08**	0.00	0.03
Discus	**0.07**	0.00	0.00
Pole.vault	0.00	**0.23**	0.00
Javeline	0.01	0.01	**0.11**
X1500m	0.00	**0.21**	0.00

Méthodes d'agrégation
(1/3)

Il existe plusieurs méthodes d'agrégations **additives** :

- Addition des rangs dans chaque indicateurs/variables k pour le pays ou l'individu i :

$$CI_i = \sum_{k=1}^{K} Rank_{ki}$$

- Pondération linéaire :

$$CI_i = \sum_{k=1}^{K} w_k I_{qi}$$

$$\sum_{k=1}^{K} w_k = 1 \text{ et } 0 \leq w_k \leq 1$$

Méthodes d'agrégation
(2/3)

Les méthodes d'agrégation additives impliquent certaines **limites** :

La fonction d'agrégation additive existe si et seulement si les indicateurs sont mutuellement indépendants (Debreu, 1960). Cela signifie que la contribution marginale de chaque variable est indépendante. Cela revient à faire l'hypothèse qu'aucune synergie n'est observée entre les différentes variables.

Exemple :

Score	E	S	G	ESG
Entreprise 1	15	6	9	30
Entreprise 2	10	10	10	30

L'entreprise 1 et l'entreprise 2 ont un score ESG équivalent, pourtant les sous-indices révèlent deux entreprises structurellement différentes.
La méthode additive revient à une pleine compensabilité implicite des indicateurs.

Méthodes d'agrégation
(3/3)

Les méthodes dites d'agrégation géométrique permettent d'apporter une solution intermédiaire :

$$CI_i = \prod_{k=1}^{K} x_{k,i}^{w_k}$$

Score	E	S	G	ESG	Moyenne additive	Moyenne géométrique
Entreprise 1	15	6	9	30	10	9.32
Entreprise 2	10	10	10	30	10	10

Exercices

Exercice 3 :

Développer une fonction d'agrégation additive qui prend en paramètre un vecteur de poids et une matrice de variables, et qui calcule l'indice composite à partir de ces deux paramètres.
La fonction doit vérifier que les dimensions des paramètres soient égales (gestion des erreurs).

Exercice 4 :

Développer la même fonction que dans l'exercice 3, mais sous forme d'agrégation géométrique.

Exercices

Exercice 5 :

Développer une fonction (ou un ensemble de fonctions) qui permette de réaliser les étapes 1 à 3 de la pondération avec l'ACP.
Tester ensuite les résultats sur les données `data(decathlon2)`