

Projet STATIS: Réponse feuille de route

ATTOUMANI Ibrahim et MANNEQUIN Jeanne

Contents

1	Situation 1	2
1.1	Préliminaires	2
1.1.1	Produit scalaire	2
1.1.2	Coefficient RV	5
1.2	Programme de STATIS 1	7
1.2.1	Programme	7
1.2.2	Équivalence avec l'ACP d'un tableau juxtaposé "dépliant" le tableau cubique	9
1.2.3	Illustration d'un ACP d'un tableau juxtaposé dépliant le tableau cubique	15
1.2.4	Quelles ACP d'autres "dépliages" du tableau cubique ?	19
2	Situation 2	20
2.1	De nouvelles matrices dans un nouvel espace	20
2.2	STATIS 2	20
2.2.1	Graphiques directs de STATIS	20
2.2.2	Graphiques projetant les individus et les variables initiales	21
2.2.3	Aides à l'interprétation	21
3	Annexe	22
3.0.1	Programme des fonctions prd_scalaire et norme	22
3.0.2	Coefficient RV	23
3.0.3	Dépliage du tableau cubique en un tableau juxtaposé :	24

Introduction

L'ACP que nous avons l'habitude de voir est une méthode d'analyse de données qui permet de résumer les informations contenues dans un tableau de données. Dans le cas de plusieurs tableaux, il nous faut trouver une méthode permettant d'analyser les données dans leur ensemble.

On se propose d'étendre l'ACP à l'analyse d'un multi-tableau. L'objectif est de comparer les individus et les variables de plusieurs tableaux. Il existe plusieurs façons de la faire, STATIS en est une. STATIS est une méthode qui permet de synthétiser les informations contenues dans les différents tableaux en une seule représentation. Il existe aussi plusieurs types de multi-tableaux. Nous en distinguons deux : les multi-tableaux à trois entrées (situation 1) et les multi-tableaux avec partition thématique (situation 2).

- Le tableau à trois entrées est un tableau de taille $n \times p \times T$, où n est le nombre d'individus, p le nombre de variables et T les différentes dates. Les indices seront notés $1 \leq i \leq n$, $1 \leq j \leq p$ et $1 \leq t \leq T$
- Le tableau avec partition thématique est composé de q tableaux décrivant les n individus à l'aide de groupes de variables différents, chaque groupe appartenant conceptuellement à un thème précis. Chacun de ces tableaux X_m , avec $1 \leq m \leq q$, possède p_m variables (colonnes).

1 Situation 1

Dans cette première partie nous utilisons les tableaux à trois entrées. Un tel tableau peut être décomposé en T tableaux juxtaposés de dimensions identiques (n, p) . Cette identité de dimension entre les T tableaux homologues permet une extension de l'ACP dans laquelle on considère chaque tableau comme une "variable" décrivant $n \times p$ "individus" (i, j) . Nous formalisons d'abord cette extension de l'ACP, STATIS 1.

1.1 Préliminaires

Il est possible de fabriquer ou de trouver un tableau à trois entrées (INSEE, ...). R propose justement un jeu de données (simulated) du package "multiblock", composé de 4 tableaux (A, B, C et D), de chacun 200 individus et 10 variables. Ces tableaux, variables et individus ne portent pas de nom mais des lettres ou numéros.

Puisque nos valeurs sont des indicateurs numériques, on les centre-réduit.

1.1.1 Produit scalaire

La matrice diagonale des poids des n individus est W (par défaut, $W = \frac{1}{n}I_n$). On considérera également une matrice diagonale des poids des p colonnes : $C = \frac{1}{p}I_p$ par défaut.

1. Le produit scalaire entre deux matrices A et B de taille (n, p) est :

$$[A|B] = \text{tr}(CA'WB)$$

La norme d'une matrice A correspondant à ce produit scalaire sera notée $[|A|]$.

- a) Ecrivons le produit scalaire sous forme $\text{tr}(\tilde{A}'\tilde{B})$ (produit scalaire de Frobenius) en explicitant la transformation Z vers \tilde{Z} , $\forall Z (n, p)$.

Soit $A, B \in M_{n,p}(\mathbb{R})$

$$\begin{aligned}[A | B] &= \text{tr}(CA'WB) \\ &= \text{tr}(C^{\frac{1}{2}}A'W^{\frac{1}{2}}W^{\frac{1}{2}}BC^{\frac{1}{2}}) \\ &= \text{tr}((W^{\frac{1}{2}}AC^{\frac{1}{2}})'W^{\frac{1}{2}}BC^{\frac{1}{2}}) \\ &= \text{tr}(\tilde{A}'\tilde{B})\end{aligned}$$

où $\forall (n, p) \quad \tilde{Z} = W^{\frac{1}{2}} Z C^{\frac{1}{2}}$

b) Pour montrer que $[A \mid B] = \text{tr}(\tilde{A}' \tilde{B})$ est un produit scalaire, nous devons démontrer les propriétés suivantes :

(i) **Symétrie** : $\forall A, B \in M_{n,p}(\mathbb{R})$

$$[A \mid B] = [B \mid A]$$

(ii) **Linéarité** : $\forall A, B_1, B_2 \in M_{n,p}(\mathbb{R})$ et $\alpha, \beta \in \mathbb{R}$

$$[A \mid (\alpha B_1 + \beta B_2)] = \alpha [A \mid B_1] + \beta [A \mid B_2]$$

(iii) **Positivité définie** : $\forall A \in M_{n,p}(\mathbb{R})$

$$[A \mid A] \geq 0 \quad \text{et} \quad [A \mid A] = 0 \iff A = 0.$$

Vérifions ces propriétés :

(i) Soit $A, B \in M_{n,p}(\mathbb{R})$

$$\begin{aligned} [A \mid B] &= \text{tr}(\tilde{A}' \tilde{B}) \\ &= \text{tr}((W^{\frac{1}{2}} A C^{\frac{1}{2}})' W^{\frac{1}{2}} B C^{\frac{1}{2}}) \\ &= \text{tr}(C^{\frac{1}{2}} A' W^{\frac{1}{2}} W^{\frac{1}{2}} B C^{\frac{1}{2}}) \end{aligned}$$

Or, par la propriété de la trace, $\text{tr}(AB) = \text{tr}(BA)$ et l'invariance par transposition des matrices de poids $W' = W$ et $C' = C$, on a alors:

$$\begin{aligned} \text{tr}(C^{\frac{1}{2}} A' W^{\frac{1}{2}} W^{\frac{1}{2}} B C^{\frac{1}{2}}) &= \text{tr}(C^{\frac{1}{2}} B' W^{\frac{1}{2}} W^{\frac{1}{2}} A C^{\frac{1}{2}}) \\ &= [B \mid A] \end{aligned}$$

(ii) Soit $A, B_1, B_2 \in M_{n,p}(\mathbb{R})$ et $\alpha, \beta \in \mathbb{R}$

$$\begin{aligned} [A \mid (\alpha B_1 + \beta B_2)] &= \text{tr}(C^{\frac{1}{2}} A' W^{\frac{1}{2}} W^{\frac{1}{2}} (\alpha B_1 + \beta B_2) C^{\frac{1}{2}}) \\ &= \alpha \text{tr}(C^{\frac{1}{2}} A' W^{\frac{1}{2}} W^{\frac{1}{2}} B_1 C^{\frac{1}{2}}) + \beta \text{tr}(C^{\frac{1}{2}} A' W^{\frac{1}{2}} W^{\frac{1}{2}} B_2 C^{\frac{1}{2}}) \\ &= \alpha \text{tr}(\tilde{A}' \tilde{B}_1) + \beta \text{tr}(\tilde{A}' \tilde{B}_2) \\ &= \alpha [A \mid B_1] + \beta [A \mid B_2] \end{aligned}$$

La linéarité de la trace assure que cette propriété est respectée.

(iii) Soit $A \in M_{n,p}(\mathbb{R})$, on note (\tilde{a}_{ij}) le terme générique de la matrice \tilde{A} . On pose $\Delta = \tilde{A}' \tilde{A}$ et on note (δ_{ij}) le terme générique de la matrice Δ .

$$\begin{aligned} [A \mid A] &= \text{tr}(C^{\frac{1}{2}} A' W^{\frac{1}{2}} W^{\frac{1}{2}} A C^{\frac{1}{2}}) \\ &= \text{tr}(\tilde{A}' \tilde{A}) \\ &= \text{tr}(\Delta) \\ &= \sum_{i=1}^p \delta_{ii} \end{aligned}$$

Or, $\delta_{ij} = \sum_{k=1}^p \tilde{a}_{ki} \tilde{a}_{kj}$ donc:

$$\begin{aligned} \text{tr}(\tilde{A}'\tilde{A}) &= \text{tr}(\Delta) \\ &= \sum_{i=1}^p \sum_{k=1}^n \tilde{a}_{ki} \tilde{a}_{ki} \\ &= \sum_{i=1}^p \sum_{k=1}^n \tilde{a}_{ki}^2 \geq 0 \end{aligned}$$

On suppose que $[A \mid A] = 0$ montrons que $A = 0_{\mathbb{R}^{n \times p}}$:

$$\begin{aligned} [A \mid A] &= 0 \\ \iff \text{tr}(\tilde{A}'\tilde{A}) &= 0 \\ \iff \text{tr}(\Delta) &= 0 \\ \iff \sum_{i=1}^p \delta_{ii} &= 0 \\ \iff \sum_{i=1}^p \sum_{k=1}^n \tilde{a}_{ki}^2 &= 0 \end{aligned}$$

Cela implique que

$$\forall (i, k) \in [1, p] \times [1, n], \quad \tilde{a}_{ki}^2 = 0$$

donc

$$\forall (i, k) \in [1, p] \times [1, n], \quad \tilde{a}_{ki} = 0$$

donc

$$\tilde{A} = W^{\frac{1}{2}} A C^{\frac{1}{2}} = 0_{\mathbb{R}^{n \times p}}$$

On en déduit que $A = 0_{\mathbb{R}^{n \times p}}$ car W et C sont régulières donc inversibles.

Par conséquent, la quantité $[A \mid B] = \text{tr}(\tilde{A}'\tilde{B})$ est un produit scalaire, car elle satisfait toutes les propriétés requises.

- c) Écrivons le produit scalaire précédent sous forme de double somme. Soit $A, B \in M_{n,p}(\mathbb{R})$ on note (\tilde{a}_{ij}) respectivement (\tilde{b}_{ij}) le terme générique de la matrice \tilde{A} respectivement \tilde{B} . On pose $\Omega = \tilde{A}'\tilde{B}$ et on note (ω_{ij}) le terme générique de la matrice de Ω . Ainsi:

$$\begin{aligned} [A \mid B] &= \text{tr}(\tilde{A}'\tilde{B}) \\ &= \text{tr}(\Omega) \\ &= \sum_{i=1}^p \omega_{ii} \\ &= \sum_{i=1}^p \sum_{k=1}^n \tilde{a}_{ki} \tilde{b}_{ki} \end{aligned}$$

Or, $\forall A, \tilde{A}$ a pour élément générique $\tilde{a}_{ki} = \sqrt{w_k} \sqrt{c_i} a_{ki}$ donc :

$$[A \mid B] = \sum_{k=1}^n \sum_{i=1}^p w_k c_i a_k i b_{ki}$$

- d) La fonction “`prd_scalaire`” est définie en Annexe. Toutefois, pour mieux comprendre son utilisation, nous présentons ci-dessous un exemple concret de son application. Le programme du précédent produit scalaire est affiché, accompagné de la norme associée à ce produit scalaire.

La fonction calculant le produit scalaire de Frobénius

Considérons à présent un exemple d’application de ma fonction **`prd_scalaire`** (le produit scalaire de Frobénius) ci-dessous :

Nous commençons par initialiser les matrices A et B avec les valeurs suivantes :

$$A = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \\ 0 & 2 & -1 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 & 1 & 2 \\ -1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

Maintenant que les matrices A et B sont initialisées, nous pouvons appliquer la fonction `prd_scalaire` pour calculer le produit scalaire de Frobénius entre elles.

```
# Initialisation de A et B
a <- c(0,1,1,0,-1,0,1,2,0,-1,0,-1)
b <- c(-1,-1,1,0,1,0,0,1,2,1,0,-1)
A = matrix(a, nrow =4)
B = matrix(b, nrow =4)

# On applique la fonction prd_scalaire
resultat = prd_scalaire(A,B)
```

Suite à l’application de la fonction `prd_scalaire` sur les matrices A et B , le résultat obtenu est 0.08333333, donc $[A \mid B] = 0.08333333$.

Après avoir calculé le produit scalaire de Frobénius entre les matrices A et B , nous allons maintenant calculer la norme associée à ce produit scalaire ($[\|\cdot\|] = \sqrt{[\cdot \mid \cdot]}$) pour chaque matrice, A et B . La fonction pour calculer la norme associée à ce produit est définie en Annexe.

```
# On utilise les matrices précédente afin de calculer leurs normes
rn1 = norme(A)
rn2 = norme(B)
```

Nous obtenons après calcul les résultats suivants: $[\|A\|] = 0.9128709$ et $[\|B\|] = 0.9574271$

1.1.2 Coefficient RV

On définit le coefficient de RV d’Escoufier entre deux matrices A et B de taille (n, p) par :

$$R(A, B) = \frac{[A \mid B]}{[\|A\|] [\|B\|]}$$

- a) Géométriquement, le coefficient de RV représente le cosinus entre les matrices A et B .

- b) Les fonctions pour calculer le coefficient de RV et fournir la matrice des coefficients de RV en (n, p) tableaux $X_{(n,p)}$ seront présentées en Annexe. En revanche, ici, nous expliciterons un exemple d'application de celles-ci.

Supposons que nos matrices X_1, X_2, X_3 et X_4 sont toutes de dimension 3×2 :

$$X_1 = \begin{pmatrix} 2 & -1 \\ 0 & -2 \\ 1 & 1 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 1 & -2 \\ -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X_3 = \begin{pmatrix} -1 & 0 \\ 2 & 1 \\ 0 & -1 \end{pmatrix}, \quad X_4 = \begin{pmatrix} 0 & -1 \\ 1 & 2 \\ -2 & 0 \end{pmatrix}$$

$$[X_1|X_2] = \text{tr}(CX_1'W_BX_2)$$

$$\frac{1}{6} \begin{pmatrix} 2 & 0 & 1 \\ -1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ -1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 2 & -3 \\ 1 & 3 \end{pmatrix}$$

donc $[X_1|X_2] = \frac{5}{6}$. Or, $\|X_1\| = \frac{11}{6}$ et $\|X_2\| = \frac{7}{6}$, donc $R(X_1, X_2) = \frac{[X_1|X_2]}{\|X_1\|\|X_2\|} = \frac{\frac{5}{6}}{\frac{11}{6} \cdot \frac{7}{6}} = \frac{30}{77}$

De manière analogue, on obtient les résultats suivants:

- $[X_1|X_3] = \frac{-5}{6}$
- $[X_1|X_4] = \frac{-5}{6}$
- $[X_2|X_3] = \frac{-4}{6}$
- $[X_2|X_4] = \frac{1}{6}$
- $[X_3|X_4] = \frac{11}{6}$
- $\|X_3\| = \frac{7}{6}$
- $\|X_4\| = \frac{10}{6}$

En notant R_{ij} le terme général de la matrice R des coefficients RV, on a:

$$R_{ij} = \frac{[X_i | X_j]}{\|X_i\| \|X_j\|}$$

On en déduit alors que:

$$R = \begin{pmatrix} 1 & \frac{30}{77} & \frac{-30}{77} & \frac{-30}{110} \\ \frac{30}{77} & 1 & \frac{-24}{49} & \frac{6}{70} \\ \frac{-30}{77} & \frac{-24}{49} & 1 & \frac{24}{70} \\ \frac{-30}{110} & \frac{6}{70} & \frac{24}{70} & 1 \end{pmatrix}$$

Pour appliquer la fonction `coeff_RV`, nous utiliserons les données citées en introduction, à savoir les données “simulated” du package “multiblock”.

```
# Tableau à trois dimension
data(simulated)

A = simulated$A
B = simulated$B
C = simulated$C
D = simulated$D

n_tableaux = list(A,B,C,D)

resultats_n = coef_RV(n_tableaux)
```

Le résultat obtenu est la matrice suivante arrondie à 10^{-4} :

$$\begin{pmatrix} 1.0000 & 0.0335 & -0.1629 & -0.1365 \\ 0.0335 & 1.0000 & 0.2213 & -0.0323 \\ -0.1629 & 0.2213 & 1.0000 & 0.0724 \\ -0.1365 & -0.0323 & 0.0724 & 1.0000 \end{pmatrix}$$

1.2 Programme de STATIS 1

1.2.1 Programme

- a) Dans le contexte de l'ACP des tableaux juxtaposés, les “variables” sont les différents tableaux \mathbf{X}_t , et les “individus” sont les observations à chaque période de temps. L'expression $\left[\sum_{t=1}^T \frac{u_t}{\|X_t\|} X_t \right]^2$ représente l'inertie le long d'un axe $\langle u \rangle$, où u est un vecteur I -unitaire: $\|u\|^2 = 1$. On cherche à projeter I -orthogonalement le nuage direct sur un espace $E_k = \langle F_1, \dots, F_k \rangle$ de dimension $k < p$ où F_k est la $k^{\text{ème}}$ composante principale. cela revient à maximiser l'inertie du nuage direct projeté sur $\langle u \rangle$.

$$\begin{aligned} \max_{\|u\|^2=1} \left[\sum_{t=1}^T \frac{X_t}{\|X_t\|} u_t \right]^2 &= \max_{\|u\|^2=1} \sum_{t=1}^T \sum_{\tau=1}^T u_\tau \frac{[X_\tau | X_t]}{\|X_t\| \|X_\tau\|} u_t \\ &= \max_{\|u\|^2=1} u' R u \end{aligned}$$

où:

1. X_t : $n \times p$ (la matrice X_t a des dimensions $n \times p$).
2. $\frac{u_t}{\|X_t\|} X_t$: $n \times p$ (chaque colonne de X_t est pondérée par $\frac{u_t}{\|X_t\|}$).
3. $\sum_{t=1}^T \frac{u_t}{\|X_t\|} X_t$: $n \times p$ (somme des termes précédents sur t).
4. $\left(\sum_{t=1}^T \frac{u_t}{\|X_t\|} X_t \right)'$: $p \times n$ (transposée de la matrice résultante).
5. $\left[\sum_{t=1}^T \frac{u_t}{\|X_t\|} X_t \right]^2$: 1×1 (la norme euclidienne au carré est un scalaire).
6. R est la matrice des coefficients RV

- b) Résolvons le programme ci-dessus, le langrangien associé s'écrit:

$$L(u, \lambda) = u' R u - \lambda (\|u\|^2 - 1)$$

On a alors :

$$\begin{cases} \frac{\partial L}{\partial u}(u, \lambda) = 2Ru - 2\lambda u \\ \frac{\partial L}{\partial \lambda}(u, \lambda) = \|u\|^2 - 1 \end{cases}$$

Les conditions de premier ordre donnent:

$$\iff (S) \begin{cases} Ru = \lambda u & (*) \\ \|u\|^2 = 1 & (**) \end{cases}$$

On a par suite: $u' \quad (*) \text{ et } (**) \Rightarrow u' R u = \lambda$.

D'après, (S) on en déduit que les vecteurs u solution de premier ordre sont les vecteurs propres de la matrice $R = ((R(X_t, X_\tau)))_{t, \tau}$ des coefficients RV d'Escoufier entre T tableaux $X_t(n, p)$. Or, pour tout vecteur propre u de R (tel que $\|u\| = 1$) de valeur propre λ , on a $u' R u = \lambda$.

La valeur maximale de $u'Ru$ est obtenue pour les vecteurs propres associés à la plus grande valeur propre de R .

c) Nous allons écrire le programme R fournissant les vecteurs u solutions des équations du premier ordre.

Considérons un exemple d'application de la fonction `vecval_prop`:

```
# on applique vecval_prop aux données simulated
vecval = vecval_prop(n_tableaux)

val_prop <- vecval$val_prop
vec_prop <- vecval$vec_prop
```

Après exécution de la fonction `vecval_prop` les résultats suivants:

$$U = \begin{pmatrix} -0.4935 & -0.4719 & 0.5853 & 0.4371 \\ 0.4170 & -0.6548 & 0.2544 & -0.5768 \\ 0.6774 & -0.2244 & -0.1247 & 0.6894 \\ 0.3517 & 0.5461 & 0.7597 & -0.0305 \end{pmatrix}$$

$$\lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix} = \begin{pmatrix} 1.2926 \\ 1.1270 \\ 0.8721 \\ 0.7083 \end{pmatrix}$$

où, U est la matrice des vecteurs propres et λ les valeurs propres associées arrondie à 10^{-4} .

Montrons à présent que les vecteurs u obtenus forment une base I -orthonormée.

Reprenons la matrice R des coefficients RV précédente. Soient u_i et u_j des vecteurs propres de R associés aux valeurs propres λ_i et λ_j distinctes. Comme u_i et u_j sont des solutions du problème suivant :

$$\max_{\|u\|^2=1} u'Ru$$

on a alors $\|u_i\|^2 = 1$ et $\|u_j\|^2 = 1$. Il suffit de montrer que u_i et u_j sont orthogonaux et ainsi de conclure que les vecteurs u forment une base I -orthonormée. En d'autres termes, nous allons montrer que $u_i'u_j = 0$.

Sachant que u_i et u_j sont des vecteurs propres de R associés aux valeurs propres λ_i et λ_j , nous avons :

$$\begin{cases} Ru_i = \lambda_i u_i \\ Ru_j = \lambda_j u_j \end{cases}$$

Par la suite, on a:

$$\begin{aligned} u_i'Ru_j &= \lambda_j u_i'u_j \\ \iff u_i'R'u_j &= \lambda_j u_i'u_j \quad (\text{car } R \text{ est une matrice symétrique}) \\ \iff (Ru_i)'u_j &= \lambda_j u_i'u_j \\ \iff (\lambda_i u_i)'u_j &= \lambda_j u_i'u_j \\ \iff \lambda_i u_i'u_j &= \lambda_j u_i'u_j \\ \iff (\lambda_i - \lambda_j)u_i'u_j &= 0 \\ \iff u_i'u_j &= 0 \quad (\text{car } \lambda_i \neq \lambda_j) \end{aligned}$$

On en déduit donc que les vecteurs u forment une base I-orthonormée.

La fonction “**vecval_prop()**” ci-dessus nous donne les vecteurs propres u_i de la matrice des coefficients RV R associée aux valeurs propres λ_i .

1.2.2 Équivalence avec l’ACP d’un tableau juxtaposé “dépliant” le tableau cubique

a) Il suffit de décomposer X_t comme une juxtaposition de colonnes:

$$X_t = [x_t^1 \dots x_t^p]_{\text{Profils Colonnes}} = \begin{bmatrix} x_{t,1} \\ \vdots \\ x_{t,n} \end{bmatrix}_{\text{Profils Lignes}}$$

et de le réécrire sous forme “verticalisée”:

$$y^t = \begin{pmatrix} x_t^1 \\ \vdots \\ x_t^p \end{pmatrix} \in \mathbb{R}^{np} \quad \text{où} \quad \forall j \in [1, p], x_t^j = \begin{pmatrix} x_{t,1}^j \\ \vdots \\ x_{t,n}^j \end{pmatrix}$$

Avant de démontrer l’égalité des produits scalaires, définissons d’abord le Produit Kronecker:

Soient A une matrice de taille $m \times n$ et B une matrice de taille $p \times q$. Leur produit tensoriel est la matrice $A \otimes B$ de taille mp par nq , définie par blocs successifs de taille $p \times q$, le bloc d’indice i, j valant $a_{ij}B$.

En d’autres termes,

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}$$

Maintenant, montrons que :

$$[X^s | X^t] = \langle y^s | y^t \rangle_L$$

où $L = C \otimes W(\text{ProduitdeKronecker})$.

Sachant que, $W = \frac{1}{n}I_n$ et $C = \frac{1}{p}I_p$ on en déduit que $L = \frac{1}{np}I_{np}$.

D’une part, soit $(s, t) \in [1, T]$:

$$\begin{aligned} [X_s | X_t] &= tr(\tilde{X}_s' \tilde{X}_t) \\ &= \sum_{k=1}^n \sum_{i=1}^p w_k c_i x_{s,k}^i x_{t,k}^i \\ &= \sum_{k=1}^n \sum_{i=1}^p \frac{1}{np} x_{s,k}^i x_{t,k}^i \end{aligned}$$

D’autre part:

$$\begin{aligned}
\langle y^s | y^t \rangle_L &= y^{s'} L y^t \\
&= \sum_{i=1}^p \frac{1}{np} x_s^{i'} x_t^i \\
&= \sum_{i=1}^p \frac{1}{np} \sum_{k=1}^n x_{s,k}^i x_{t,k}^i \\
&= \sum_{k=1}^n \sum_{i=1}^p \frac{1}{np} x_{s,k}^i x_{t,k}^i
\end{aligned}$$

Donc $[X^s | X^t] = \langle y^s | y^t \rangle_L$.

Il en découle que X^t est “verticalisé” en $y^{t*} = y^t$ L -normé.

Par suite, on note $Y = [y^1, \dots, y^T]$. Un individu correspond à une ligne de ce tableau, correspondant donc à l'indice (j, i) dans le cube initial. Un individu est un vecteur de taille T , dont les coordonnées sont les T valeurs de la case (i, j) du tableau X_t sur la période. C'est la trajectoire de cette case. Son poids est $L(j, i) = w_i c_j = \frac{1}{np}$.

Par exemple, pour un tableau cubique ventilant la valeur ajoutée d'une économie par région (i) , secteur (j) et année (t) , l'individu du tableau dépliant ainsi le cube serait la trajectoire dans le temps de la V.A. réalisée par le secteur j dans la région i .

Et on a :

$$\left[\left| \sum_{t=1}^T u_t \frac{X_t}{\|X_t\|} \right| \right]^2 = \|Y^* u\|_W^2 = \|F\|_W^2$$

, où $F = Y^* u$ est la composante associée à u .

- b) Pour déduire que les composantes principales F^1, \dots, F^k, \dots sont orthogonales au sens du produit scalaire convenable, nous devons considérer la façon dont ces composantes sont construites à partir des vecteurs propres obtenus.

$$\begin{aligned}
\langle F_k | F_l \rangle_L &= F_k' L F_l \\
&= u_k' Y^{*'} L Y^* u_l \\
&= 1 \quad \text{si } k = l \\
&= 0 \quad \text{si } k \neq l
\end{aligned}$$

- c) Dépliage du tableau cubique en un tableau juxtaposé :

Composantes principales des tableaux dépliés :

On veut donner la fonction des composantes principales correspondant aux vecteurs u (vec_prop)
 composantes_principales = **function**(u, X_t){

```

  p = length(X_t[1,]) # la dimension des individus
  n = length(X_t[,1]) # la dimension des variables
  W = diag(n)/n
  M = matrix(0, nrow = p, ncol = p)
```

```

for (i in 1:p) {
  M[i,i] = 1/sqrt(t(X_t[,i]) %*% W %*% X_t[,i])
}

F <- X_t %*% M %*% u # calcul de la matrice des composantes principales

return(F)
}
comp_prin = composantes_principales(vec_prop_jux, mat_jux)
print(comp_prin[1:8,1:3]) # Affichage de la première composante principale

##           [,1]      [,2]      [,3]
## [1,] -1.3350171  0.7284918  0.28498025
## [2,]  2.6149460 -0.7778572 -0.44579708
## [3,] -0.6205920 -1.0449091 -0.09381266
## [4,]  0.5367399 -0.2626285 -0.39001332
## [5,]  0.4548871 -0.5489805  0.83000712
## [6,]  0.1119155 -0.7373719 -0.33002573
## [7,] -0.7433038  0.6872098  0.26976932
## [8,]  0.6082812 -2.0817439  0.46856410

# On veut maintenant programmer la fonction calculant la k-ième composante F_k ainsi que la
# composante normée f_k correspondante
composante_k = function(u, X_t, k){

  # calcul des composantes principales
  comp_prin = composantes_principales(u, X_t)
  F_k = comp_prin[,k]

  f_k = F_k / sqrt(sum(F_k^2)) # calcul de la k-ième composante principale normée

  return(list(F_k = F_k, f_k = f_k))
}

pc1 <- composante_k(vec_prop, mat_jux, 1)$F_k
pc1_nr <- composante_k(vec_prop, mat_jux, 1)$f_k

# Affichage des premières lignes de la première composante principale et de sa version normée
cat("Extrait de la 1ère composante :\n")

## Extrait de la 1ère composante :
print(head(pc1))

## [1] -1.3350171  2.6149460 -0.6205920  0.5367399  0.4548871  0.1119155
cat("\nExtrait de la 1ère composante normée :\n")

##
## Extrait de la 1ère composante normée :
print(head(pc1_nr))

## [1] -0.026256913  0.051430359 -0.012205709  0.010556519  0.008946650
## [6]  0.002201137

```

```

# Fonction donnant la représentation graphique des individus en plan principal (k,l)
representation_graph_ind = function(comp_p, val_prop_dep, k, l,
                                   aff_noms = FALSE, col = "navy") {
  # Créer un data frame avec les composantes principales
  df <- data.frame(F_k = comp_p[, 1], F_l = comp_p[, 2])
  inert_k = val_prop_dep[k] / sum(val_prop_dep) * 100
  inert_l = val_prop_dep[l] / sum(val_prop_dep) * 100

  # Récupérer les noms de lignes ou les numéros de ligne
  row_names <- if (is.null(rownames(df))) 1:nrow(df) else rownames(df)

  # Trouver les limites des axes x et y pour centrer l'origine
  x_limits <- range(df$F_k)
  y_limits <- range(df$F_l)
  limits <- range(c(x_limits, y_limits))

  # Plot les composantes principales dans un plan avec l'origine au centre
  plot(df$F_k, df$F_l, xlab = paste("F", k, " (", round(inert_k, 2), "%)",
                                   ylab = paste("F", l, " (", round(inert_l, 2), "%)",
                                   main = paste("Représentation des individus dans le plan (", k, ",", l, ")"),
                                   xlim = limits, ylim = limits)

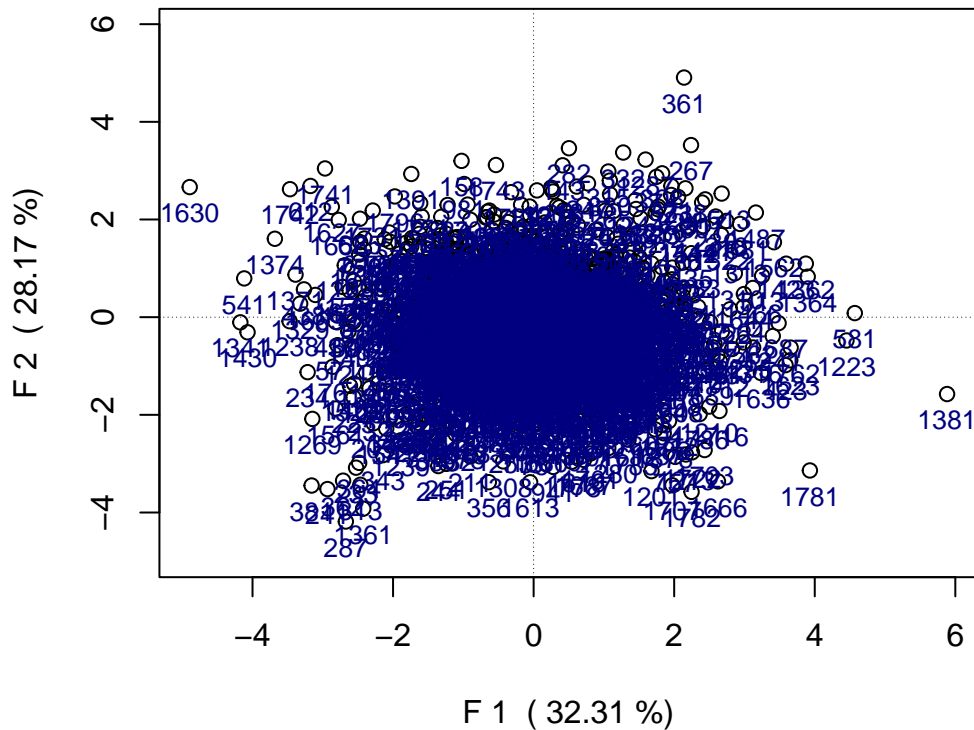
  # Ajouter des lignes en pointillé pour les axes horizontal et vertical
  abline(h = 0, lty = "dotted", lwd = 0.5)
  abline(v = 0, lty = "dotted", lwd = 0.5)

  # Afficher les noms de lignes ou les numéros de ligne si demandé
  if (aff_noms) {
    text(df$F_k, df$F_l, labels = row_names, pos = 1, cex = 0.8, col = col)
  }
}

# Exemple d'utilisation sans afficher les noms de lignes ou les numéros de ligne
representation_graph_ind(comp_prin, val_prop_jux, 1, 2)
# Exemple d'utilisation avec l'affichage des noms de lignes ou des numéros de ligne
representation_graph_ind(comp_prin, val_prop_jux, 1, 2, aff_noms = TRUE)

```

Représentation des individus dans le plan (1 , 2)



d) Cosinus entre les tableaux et les composantes principales :

```
# Calcul des cosinus entre Xt (resultats_n) et chaque composante principale F_k
cosinus = function(X_t, f_k){
  mat_cos = matrix(NA, nrow = ncol(f_k), ncol = ncol(f_k))
  for (k in 1:ncol(f_k)){
    for (j in 1:ncol(f_k)) {
      # Calcul du cosinus entre Xt et la kième composante F_k
      mat_cos[k,j] = sum(X_t[,j] * f_k[,k]) / (sqrt(sum(X_t[,j]^2)) *
                                                sqrt(sum(f_k[,k]^2)))
    }
  }
  rownames(mat_cos) <- paste("v", 1:ncol(mat_cos), sep = "")
  return(t(mat_cos))
}

# Affichage du produit scalaire entre Xt et la ième composante principale
cosinus(mat_jux, composantes_principales(vec_prop_jux, mat_jux))
```

```
##          v1          v2          v3          v4
## [1,] -0.5611042 -0.5010173  0.5466167  0.36790472
## [2,]  0.4740911 -0.6950906  0.2376074 -0.48541671
## [3,]  0.7701035 -0.2381741 -0.1164127  0.58022566
```

```
## [4,] 0.3998963 0.5797305 0.7094618 -0.02567956
```

```
representation_graph_var = function(vec_prop, X_t, val_prop, k, l){
  inert_k = val_prop[k]/sum(val_prop) * 100 #l'inertie captée par la comp k
  inert_l = val_prop[l]/sum(val_prop) * 100 #l'inertie captée par la comp k

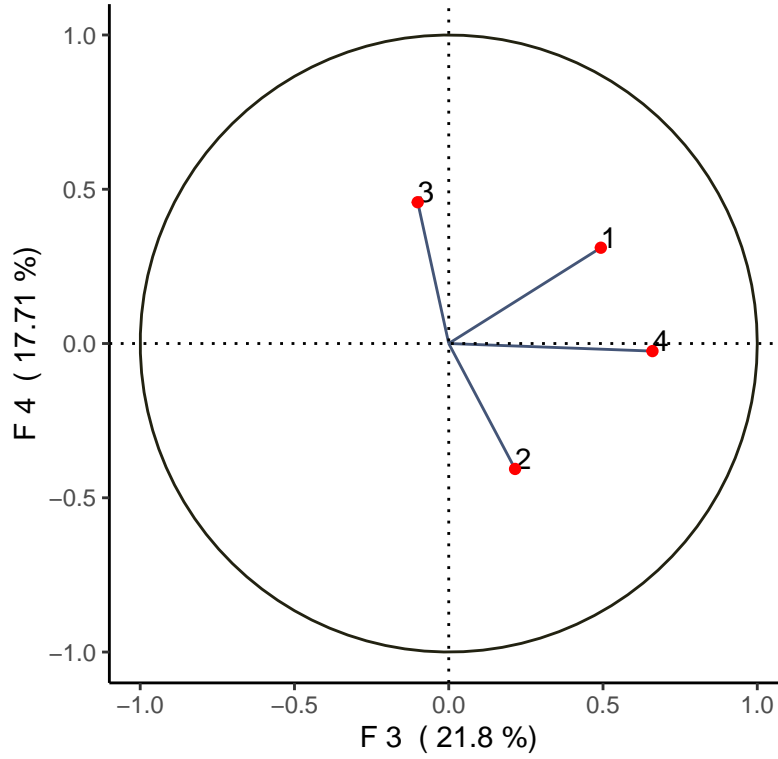
  mat_cos = data.frame(cosinus(X_t, composantes_principales(vec_prop, X_t)))
  F_k = mat_cos[,k] # Récupération de la kième composante principale
  F_l = mat_cos[,l] # Récupération de la lième composante principale

  # Création des données pour le cercle unité
  df <- data.frame(
    varabs = cos(seq(0, 2 * pi, length.out = 100)),
    varord = sin(seq(0, 2 * pi, length.out = 100))
  )

  # Créer le graphique en utilisant ggplot2
  ggplot(mat_cos, aes(x = F_k, y = F_l)) +
    geom_segment(aes(xend = 0, yend = 0), color = "#445577") +
    geom_text(aes(label = rownames(mat_cos)), hjust = 0, vjust = 0) +
    geom_point(color = "red", size = 1.5) +
    theme_classic() +
    coord_fixed(ratio = 1) +
    geom_hline(yintercept = 0, linetype = "dotted", lwd = 0.5) +
    geom_vline(xintercept = 0, linetype = "dotted", lwd = 0.5) +
    labs(x = paste("F", k, " (", round(inert_k, 2), "%)"),
         y = paste("F", l, " (", round(inert_l, 2), "%)")) +
    ggtitle(paste("Représentation des variables dans le plan (", k, ",", l, ")")) +
    geom_path(data = df, aes(varabs, varord), color = "#222211", linewidth = 0.5) + # Ajout du cercle
    xlim(-1, 1) +
    ylim(-1, 1)
}

# Affichage de la représentation graphique des variables en plan principal (1,2)
representation_graph_var(vec_prop_jux, mat_sym, val_prop_jux, 3, 4)
```

Représentation des variables dans le plan (3 , 4)



1.2.3 Illustration d'un ACP d'un tableau juxtaposé dépliant le tableau cubique

Soit A , B et C les matrices données comme suit:

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ 2 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & -1 \\ -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 3 & 1 \\ 1 & -2 \\ 0 & -3 \end{pmatrix}$$

En utilisant le produit scalaire défini par:

$$\begin{aligned} [A \mid B] &= \text{tr}(CA'WB) \\ &= \text{tr}(C^{\frac{1}{2}}A'W^{\frac{1}{2}}W^{\frac{1}{2}}BC^{\frac{1}{2}}) \\ &= \text{tr}((W^{\frac{1}{2}}AC^{\frac{1}{2}})'W^{\frac{1}{2}}BC^{\frac{1}{2}}) \\ &= \text{tr}(\tilde{A}'\tilde{B}) \end{aligned}$$

avec $W := \frac{1}{n}I_n$ la matrice de poids des individus et $C := \frac{1}{p}I_p$ la matrice de poids des variables.

Nous allons calculer la matrice des coefficients RV d'Escoufier que l'on notera Γ . Ci-dessous la formule permettant de calculer le coefficient RV entre deux matrices A et B de taille n, p ici $n = 3$ et $p = 2$:

$$R(A, B) = \frac{[A \mid B]}{[A][B]}$$

Calculons les coefficients $R(A, B)$, $R(A, C)$ et $R(B, C)$.

Procédons étape par étape:

$$[A | B] = \text{tr}(CA^tWB) = \frac{1}{6}\text{tr}(A^tB)$$

Or,

$$\frac{1}{6} \begin{pmatrix} 1 & -1 & 2 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 3 & 1 \\ -1 & 2 \end{pmatrix}$$

donc $[A | B] = \frac{5}{6}$

Pour $[A | C]$:

$$[A | C] = \frac{1}{6}\text{tr}(A^TC)$$

Or,

$$\frac{1}{6} \begin{pmatrix} 1 & -1 & 2 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & -2 \\ 0 & -3 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 2 & -3 \\ 1 & -8 \end{pmatrix}$$

donc $[A | C] = \frac{-6}{6} = -1$

Pour $[B | C]$:

$$[B | C] = \frac{1}{6}\text{tr}(B^TC)$$

Or,

$$\frac{1}{6} \begin{pmatrix} 2 & -1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & -2 \\ 0 & -3 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 5 & 4 \\ -3 & -4 \end{pmatrix}$$

donc $[B | C] = \frac{1}{6}$. En utilisant la norme associée à ce produit scalaire on a:

$$[|A|] = \sqrt{\frac{11}{6}}, [|B|] = \sqrt{\frac{7}{6}} \text{ et } [|C|] = \sqrt{\frac{24}{6}} = 2.$$

D'où

$$R(A, B) = \frac{\frac{5}{6}}{\sqrt{\frac{11}{6}}\sqrt{\frac{7}{6}}}$$

$$R(A, C) = \frac{-1}{2\sqrt{\frac{11}{6}}}$$

et

$$R(B, C) = \frac{\frac{1}{6}}{2\sqrt{\frac{7}{6}}}$$

Par symétrie du produit scalaire, la matrice Γ des coefficients RV est donc égale à:

$$\Gamma = \begin{pmatrix} 1.0000000 & 0.56980288 & -0.36927447 \\ 0.5698029 & 1.00000000 & 0.07715167 \\ -0.3692745 & 0.07715167 & 1.00000000 \end{pmatrix}$$

Maintenant, considérons le dépliement des matrices juxtaposées :

$$X = \begin{pmatrix} 1 & 2 & 3 \\ -1 & -1 & 1 \\ 2 & 0 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -2 \\ 2 & 1 & -3 \end{pmatrix}$$

où la première variable est la matrice A dépliée, la deuxième variable est la matrice B dépliée et la troisième variable est la matrice C dépliée.

On pose M la matrice diagonale p, p (ici $p = 3, n = 6$) dont le coefficient $M_{ii} = \frac{1}{\|X_i\|_W}$ avec:

- $W := \frac{1}{n} I_n$
- X_i la i ème variable
- $\|X_i\|_W = \sqrt{(X_i^t W X_i)}$

Ici M sera donc égale à:

$$M = \begin{pmatrix} \frac{1}{\sqrt{\frac{11}{6}}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{\frac{7}{6}}} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

On a alors:

$$\Delta := M X^t W X M = \Gamma$$

La fonction ci-dessous fait les calculs matriciels puis nous donne la matrice obtenue delta qui est égale à la matrice des coefficients RV calculée précédemment.

```
# Données
X <- matrix(
  c( 1, 2, 3,
    -1,-1, 1,
     2, 0, 0,
     0,-1, 1,
     1, 0,-2,
     2, 1,-3
  ), nrow = 6, byrow = TRUE)

# Ajout des noms de lignes
rownames(X) <- paste0("L", 1:6)

oper_inert = function(X){

  p = length(X[1,]) # la dimension des individus
  n = length(X[,1]) # la dimension des variables
  W = diag(n)/n
  M = matrix(0, nrow = p, ncol = p)

  for (i in 1:p) {
    M[i,i] = 1/sqrt(t(X[,i])%*%W%*%X[,i])
  }
}
```

```

return(M %*% t(X) %*% W %*% X %*% M)
}

```

```

# Calcul de la matrice
Delta <- oper_inert(X)
print(Delta)

```

```

##           [,1]      [,2]      [,3]
## [1,]  1.0000000 0.56980288 -0.36927447
## [2,]  0.5698029 1.00000000  0.07715167
## [3,] -0.3692745 0.07715167  1.00000000

```

```

result_propre = eigen(Delta) # list ayant les valeurs et vecteurs propres
valp = result_propre$values #valeurs propres
vecp = result_propre$vectors

```

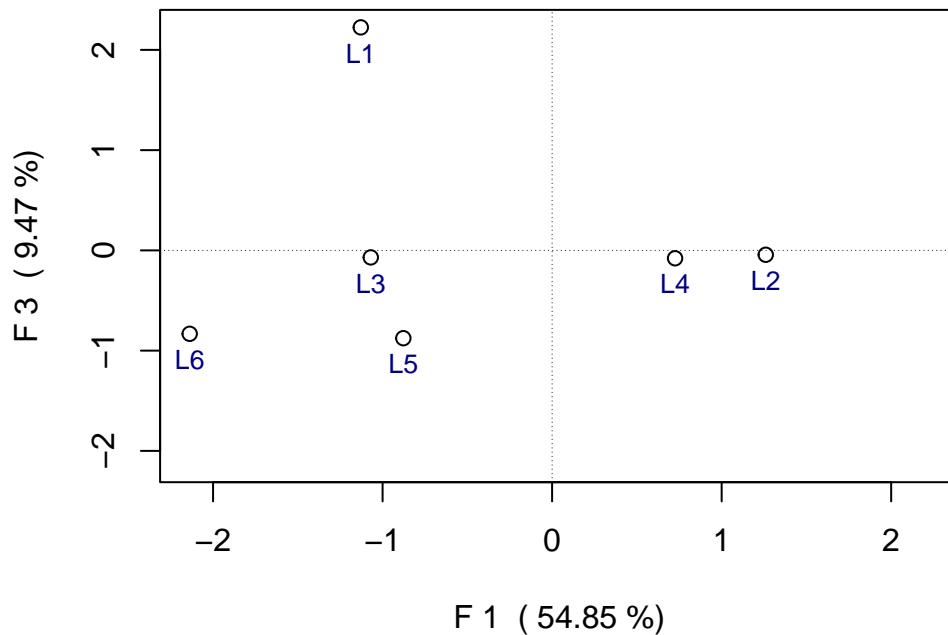
Finalement, on en déduit qu'il y a équivalence entre les matrices juxtaposées A, B et C avec la matrice X dont les variables sont les matrices A, B et C dépliées.

```

comp = composantes_principales(vecp,X)
representation_graph_ind(comp, valp, 1, 3, aff_noms = T)

```

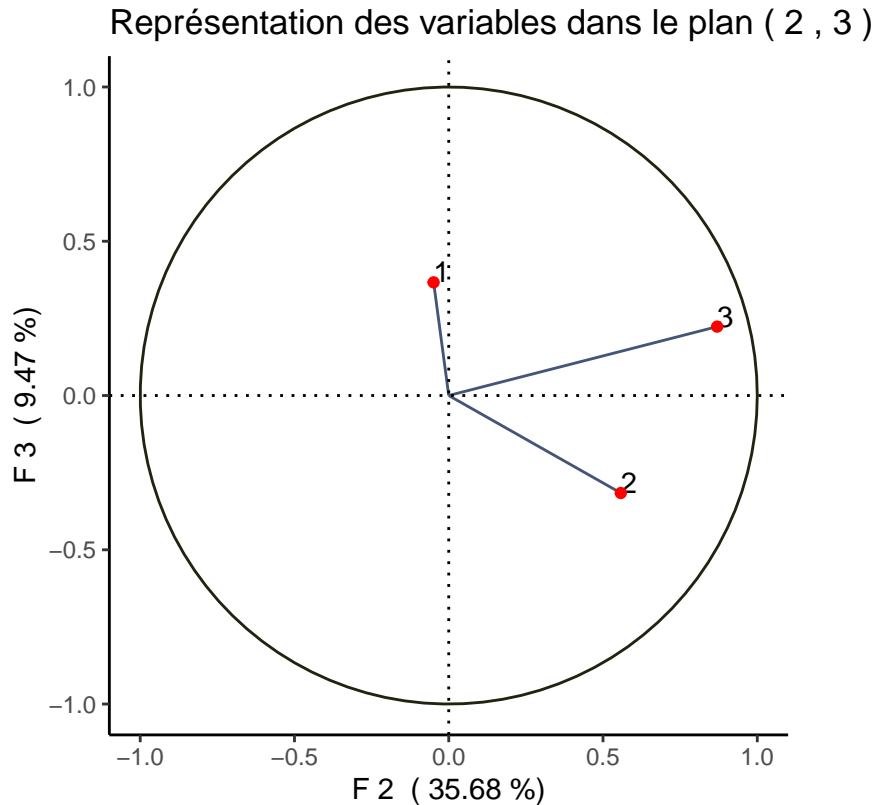
Représentation des individus dans le plan (1 , 3)



```

representation_graph_var(vecp,X,valp,2,3)

```



1.2.4 Quelles ACP d'autres "dépliage" du tableau cubique ?

- a) Disons que l'on dispose d'un tableau cubique $X(i,j,t)$. On peut le déplier de trois manières différentes : $Y((i,j),t)$ de façon à avoir les individus et caractéristiques en fonction du temps. Pour avoir les caractéristiques dans le temps pour chaque individu $Y((j,t),i)$. Ou pour avoir les individus dans le temps pour chaque caractéristique $Y((i,t),j)$.
Chacun de ces dépliages donnera une matrice différente et donc une ACP différente. Mais chacun reste intéressant.
- b) Selon le dépliage choisi, les individus et variables ne seront pas les mêmes. En individus on peut avoir :
- Les individus par caractéristiques (ou caractéristiques par individus) décrit dans le temps,
 - Les individus dans le temps (à un instant donné), décrit par les caractéristiques,
 - Les caractéristiques dans le temps (à un instant donné), décrivant les individus.
- Les ACP à envisager dépendent du jeu de données que l'on souhaite étudier. Ici les individus et variables ne sont pas clairement définis car ne porte que des numéros. Ce qui est sûr c'est que les individus regrouperont les tableaux dépliés. Les éléments à projeter en supplémentaire dépendent, encore une fois, de ce que l'on souhaite étudier.
- c) Un inconvénient est qu'on perde une grosse partie de l'information qui sera noyée dans le tableau déplié. Cependant, cela permet de mieux visualiser les données et de les traiter plus facilement (utilisation du produit matriciel possible). Le plus gros inconvénient reste la perte de la symétrie, par exemple lors d'un dépliage de $X(i,j,t)$ en $Y((i,j),t)$. En effet, les individus et variables ne sont plus équivalents. Cela peut poser problème si l'on souhaite comparer les individus et variables entre eux.

L'idéal serait de réaliser les trois dépliages possibles et de les comparer pour voir lequel est le plus pertinent !

Tout dépend du tableau initial.

2 Situation 2

2.1 De nouvelles matrices dans un nouvel espace

1.1.a) Calculons P_m en fonction de X_m et M_m :

$$P_m = X_m M_m X'_m$$

b) Les matrices P_m se trouvent dans l'espace $\mathbf{P} = \mathbf{M}_n(\mathbb{R})$ où n est le nombre d'individus. Cet espace est de taille $\frac{n(n+1)}{2}$ car $P_m = X_m M_m X'_m$ produit une matrice symétrique de taille $n \times n$ contenant $\frac{n(n+1)}{2}$ éléments indépendants.

1.2.a) Les poids naturel des lignes de P_m serait $W = \frac{1}{n} I_n$ même chose pour les colonnes.

b) On peut munir \mathbf{P} du produit scalaire de Frobenius qui est défini par:

$$(P_m, P_k) = \text{tr}(P_m W P_k W) \quad (p)$$

c) Les tableaux sont mesurés à des échelles différentes, on se doit donc de normées les tableaux P_m afin de les ramenées sur la même échelle. La normalisation permet de rendre les comparaisons moins sensibles à l'échelle des données et plus centrées sur les relations entre les individus.

d) Pour mesurer la proximité entre deux matrices P_m et P_k on utilise le produit scalaire (p) afin de calculer le coefficient RV entre P_m et P_k . Si le coefficient est proche de 1 on pourra dire que la matrice X_m et X_k sont similaire (corréées fortement) et donc une similarité plus grande entre les ensembles d'individus représentés par X_m et X_k . En revanche, si le coefficient RV est proche de -1 on dira que les matrices X_m et X_k sont anticorréées. Enfin, si le coefficient RV est proche de 0, on dira que X_m et X_k sont décorréées (on en peut pas conclure).

2.2 STATIS 2

On applique le programme de STATIS 1 à un ensemble de matrices P_m issues de tableaux thématiques décrivant les mêmes individus. On note F_1, \dots, F_k les composantes principales obtenues.

2.2.1 Graphiques directs de STATIS

a) Pour projeter chacune des matrices P_m sur un graphe dont les directions sont un couple de composantes principales (F_k, F_l) , on peut utiliser la méthode de la projection orthogonale.

Tout d'abord, nous calculons les composantes principales $(F_k$ et $F_l)$ à partir de la matrice P résultant de l'analyse STATIS 1.

Ensuite, pour chaque P_m , nous calculons les produits scalaires entre P_m et les composantes principales F_k et F_l .

Enfin, ces produits scalaires servent de coordonnées pour projeter les points correspondant à P_m sur le plan défini par les composantes principales F_k et F_l .

b)

```
# Tableau à trois dimension
```

```
data(chickenk)
```

```
M = as.matrix(chickenk$Mortality)
```

```
rownames(M) = seq(1,351,1)
```

```
FS = as.matrix(chickenk$FarmStructure)
```

```
rownames(FS) = seq(1,351,1)
```

```

OFH = as.matrix(chickenk$OnFarmHistory)
rownames(OFH) = seq(1,351,1)

FC = as.matrix(chickenk$FlockCharacteristics)
rownames(FC) = seq(1,351,1)

CTS = as.matrix(chickenk$CatchingTranspSlaught)
rownames(CTS) = seq(1,351,1)

data = list(M, FS, OFH, FC, CTS)

mat_coldiff = function(X){

  p = length(X[1,]) # la dimension des individus
  M = diag(p)/p

  Pm = X %*% M %*% t(X)

  return(Pm)

}

```

2.2.2 Graphiques projetant les individus et les variables initiales

- a) Pour montrer que les composantes principales sont des matrices symétriques, nous devons rappeler que les matrices de covariance (ou de corrélation) utilisées dans l'analyse des composantes principales (ACP) sont symétriques. Les composantes principales sont calculées à partir de ces matrices de covariance, ce qui garantit que les composantes principales elles-mêmes seront symétriques.
- b) Chaque composante principale peut être interprétée comme une matrice de produits scalaires entre individus. En utilisant ces produits scalaires, nous pouvons obtenir une image du nuage des individus dans un plan de dimension réduite en utilisant des techniques de visualisation telles que l'Analyse en Composantes Principales (PCA) ou d'autres méthodes de réduction de dimensionnalité. Une fois que nous avons cette représentation en deux dimensions des individus, nous pouvons mettre en relation cette image avec les variables des tableaux en examinant les poids des variables dans chaque composante principale. Les variables qui contribuent le plus aux composantes principales sont celles qui ont le plus d'influence sur la position des individus dans l'espace réduit.
- c)

2.2.3 Aides à l'interprétation

- a) Le cosinus carré entre une matrice P_m et une composante principale F_k peut être calculé comme suit :

$$\cos^2(P_m, F_k) = \frac{\text{Var}(P_m \cdot F_k)}{\text{Var}(P_m) \cdot \text{Var}(F_k)}$$

- b) Le calcul de $QLT_k(P_m)$, qui mesure la qualité de la représentation de P_m par rapport aux premières k composantes principales, est simplement le cosinus carré moyen de P_m par rapport à ces k composantes principales :

$$QLT_k(P_m) = \frac{1}{k} \sum_{l=1}^k \cos^2(P_m, F_l)$$

- c) Le CTR (Cumulative Total RV) de la k -ième composante principale pour P_m peut être calculée comme suit:

$$CTR_k(P_m) = 1 - QLT(P_m)$$

Ce score mesure la proportion de variance de P_m qui n'est pas expliquée par les k premières composantes principales.

- d) Tentative de programmation des fonctions (à retravailler):

```
# Définition des fonctions
cos_square <- function(P, F) {
  P_dot_F <- P * F
  var_P_dot_F <- var(P_dot_F)
  var_P <- var(P)
  var_F <- var(F)
  return(var_P_dot_F / (var_P * var_F))
}

QLT_k <- function(P, Fs) {
  k <- length(Fs)
  cos_squares <- sapply(Fs, function(F) cos_square(P, F))
  return(mean(cos_squares[1:k]))
}

CTR_k <- function(P, Fs) {
  return(1 - QLT_k(P, Fs))
}

# Appliquer les fonctions au tableau de données d'application
# Supposons que P_m représente les données projetées sur une composante principale
set.seed(123) # pour la reproductibilité des résultats
P_m <- matrix(runif(100*2), ncol=2) # Exemple de données projetées (100 individus, 2 dimensions)
Fs <- replicate(5, matrix(runif(100*2), ncol=2), simplify = FALSE) # Exemple de 5 composantes principales

for (k in 1:5) {
  cat("CTR_", k, "(P_m): ", CTR_k(P_m, Fs[1:k]), "\n")
}

## CTR_ 1 (P_m): -4.949034
## CTR_ 2 (P_m): -55.89409
## CTR_ 3 (P_m): -72.87578
## CTR_ 4 (P_m): -56.04714
## CTR_ 5 (P_m): -45.76911
```

3 Annexe

3.0.1 Programme des fonctions prd_scalaire et norme

La fonction calculant le produit scalaire de Frobenius:

```
prd_scalaire = function(A,B){
  # les dimension des matrices
  na = length(A[,1]); nb = length(B[,1])
  pa = length(A[1,]); pb = length(B[1,])
```

```

# Les matrices de ponderation
Wa = (1/na)*diag(na); Ca = (1/pa)*diag(pa)
Wb = (1/nb)*diag(na); Cb = (1/pb)*diag(pb)

# Calcul des matrices A_tild et B_tild
A_tild = sqrt(Wa) %*% A %*% sqrt(Ca)
B_tild = sqrt(Wb) %*% B %*% sqrt(Cb)

# Produit scalaire
ps = sum(diag(t(A_tild)%*%B_tild))

return(ps)
}

```

La fonction calculant la norme d'une matrice A associée au produit scalaire de Frobénius

```

norme = function(A){
  return(sqrt(prd_scalaire(A,A)))
}

```

3.0.2 Coefficient RV

La fonction calculant le coefficient RV d'Escoufier:

```

coef_RV = function(T_tableaux) {
  t = length(T_tableaux)
  mat_rv = matrix(rep(NA, t * t), nrow = t, ncol = t)

  for (i in seq_along(T_tableaux)) {
    for (j in seq_along(T_tableaux)) {
      # Stocker le résultat dans une matrice
      prd_sclr = prd_scalaire(T_tableaux[[i]], T_tableaux[[j]])
      norm_i = norme(T_tableaux[[i]])
      norm_j = norme(T_tableaux[[j]])
      mat_rv[i, j] = prd_sclr / (norm_j * norm_i)
    }
  }

  return(mat_rv)
}

```

Fonction donnant les vecteurs u solutions et les valeurs propres associées:

```

vecval_prop = function(T_tableau) {
  matrice = coef_RV(T_tableau) # on calcule la matrice contenant les coefs d'Escoufier

  resultat_propre = eigen(matrice) # list ayant les valeurs et vecteurs propres
  val_prop = resultat_propre$values #valeurs propres
  vec_prop = resultat_propre$vectors #vecteurs propres associées

  return(list(val_prop = val_prop, vec_prop = vec_prop))
}

```

3.0.3 Dépliage du tableau cubique en un tableau juxtaposé :

```
oper_inert = function(X){  
  
  p = length(X[1,]) # la dimension des individus  
  n = length(X[,1]) # la dimension des variables  
  W = diag(n)/n  
  M = matrix(0, nrow = p, ncol = p)  
  
  for (i in 1:p) {  
    M[i,i] = 1/sqrt(t(X[,i])%*%W%*%X[,i])  
  }  
  mat_d = M %*% t(X) %*% W %*% X %*% M  
  return(mat_d)  
}
```