Projet STATIS: Réponse feuille de route

ATTOUMANI Ibrahim et MANNEQUIN Jeanne

Contents

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 superposables, 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 entréees 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 décrivants 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 nxp "individus" (i,j). Nous formalisons d'abord cette extension de l'ACP, appelée 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 3 tableaux (A,B et D), de chacun 200 individus et 10 varaibles. 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 :

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned}
[A \mid (\alpha B_1 + \beta B_2)] &= \text{tr}(C^{\frac{1}{2}}A'W^{\frac{1}{2}}W^{\frac{1}{2}}(\alpha B_1 + \beta B_2)C^{\frac{1}{2}}) \\
&= \alpha\text{tr}(C^{\frac{1}{2}}A'W^{\frac{1}{2}}W^{\frac{1}{2}}B_1C^{\frac{1}{2}}) + \beta\text{tr}(C^{\frac{1}{2}}A'W^{\frac{1}{2}}W^{\frac{1}{2}}B_2C^{\frac{1}{2}}) \\
&= \alpha\text{tr}(\tilde{A}'\tilde{B}_1) + \beta\text{tr}(\tilde{A}'\tilde{B}_2) \\
&= \alpha[A \mid B_1] + \beta[A \mid B_2]
\end{aligned}$$

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.

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

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

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

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

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

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 $[A \mid B] = \mathrm{tr}(\tilde{A}'\tilde{B})$ 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 $\Gamma = \tilde{A}'\tilde{B}$ et on note (γ_{ij}) le terme générique de la matrice de Γ. Ainsi:

$$\begin{aligned} [A \mid B] &= tr(\tilde{A}'\tilde{B}) \\ &= tr(\Gamma) \\ &= \sum_{l=1}^{p} \gamma_{ll} \\ &= \sum_{l=1}^{p}\sum_{k=1}^{n} \tilde{a}_{kl}\tilde{b}_{kl} \end{aligned}$$

d) Nous pouvons observer le programme du précédent produit scalaire ci-dessous, ainsi que la norme associée à ce produit scalaire.

La fonction calculant le produit scalaire de Frobénius

```
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)
 }
```

Exemple d'application sur le produit scalaire de Frobénius

```
# On initialise des matrices au hasard
A  = matrix(-16:18, nrow =7)
B = matrix(1:35, nrow =7)

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

# Affichage du résultat
print(resultat)
```

```
## [1] 120
```

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

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

Exemple d'application sur la norme

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

# Affichage du résultat
rn1; rn2
```

```
## [1] 10.14889
```

```
## [1] 20.63977
```

1.1.2 Coefficient RV

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

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

a) En termes géométriques le coefficient RV et le cosinus de A et B.

b) Ci-dessous nous trouverons le programme calculant le coefficient RV ainsi qu'une fonction donnant la matrice des coeffients RV en T tableaux $X_{(n,p)}$.

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)
}
```

Exemple d'application sur le coefficient RV d'Escoufier T tableaux X_t(n , p)

```
# 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)
print(data.frame(resultats_n))
```

```
##            X1          X2          X3          X4
## 1  1.00000000  0.03353098 -0.16294428 -0.13648245
## 2  0.03353098  1.00000000  0.22134008 -0.03233924
## 3 -0.16294428  0.22134008  1.00000000  0.07239449
## 4 -0.13648245 -0.03233924  0.07239449  1.00000000
```

1.2 Programme de STATIS 1

1.2.1 Programme

a) L'expression $\left[\left|\sum_{t=1}^{T} \frac{u_t}{[|X_t|]} X_t\right|\right]^2$

représente l'inertie dans le contexte de l'analyse factorielle. L'inertie est également appelée somme des carrés des corrélations ou variance totale. Dans le cadre de l'analyse factorielle, cette quantité mesure la dispersion totale des données dans l'espace factoriel (l'espace ou les individus se trouvent).

L'objectif du problème d'optimisation associé est de maximiser cette inertie sous la contrainte que la norme du vecteur u est égale à 1. Cela revient à trouver la direction dans laquelle la dispersion des données est maximale. L'inertie d'un vecteur u par rapport à une matrice X est définie comme la norme au carré de la projection de X sur u. Plus précisément, $\sum_{t=1}^{T}(u^T X_t)^2$ représente la somme des inerties de tous les vecteurs X_t projetés sur u.

b) Résolvons le programme ci-dessus:

$$\begin{aligned}
\max_{[|u|]^2=1} \left[\left|\sum_{t=1}^{T} \frac{X_t}{[|X_t|]} u_t\right|\right]^2 &= \max_{[|u|]^2=1} \sum_{t=1}^{T}\sum_{\tau=1}^{T} u_\tau \frac{\operatorname{tr}\left(CX'_\tau W X_t\right)}{[|X_t|]\,[|X_\tau|]} u_t \\
&= \max_{[|u|]^2=1} \sum_{t=1}^{T}\sum_{\tau=1}^{T} u_\tau \frac{[X_\tau \mid X_t]}{[|X_t|]\,[|X_\tau|]} u_t \\
&= \max_{[|u|]^2=1} u'Ru
\end{aligned}$$

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. $\left[\left|\sum_{t=1}^{T} \frac{u_t}{[|X_t|]} X_t\right|\right]^2$: 1×1 (la norme euclidienne au carré est un scalaire).
6. R est la matrice des coefficients RV

Le langrangien associée au problème d'optimisation ci-dessus s'écrit:

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

On a alors :

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

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

En multipliant l'équation $(*)$ *par* u' on obtient grâce à l'équation $(**)$ le résultat suivant:

$$u'Ru = \lambda$$

D'après, (S) on en déduit que les vecteurs u solution du premier ordre sont les vecteurs propres de la matrice $R = \sum_{t=1}^{T}\sum_{\tau=1}^{T} \frac{tr(\tilde{X}'_t \tilde{X}_\tau)}{[|X_t|][|X_\tau|]}$ 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 $u'Ru = \lambda$.

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

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

Fonction donnant les vecteurs u solution et valeurs propres

```
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))
}
```

exemple d'application de la fonction vecval_prop

```
vecval = vecval_prop(n_tableaux)

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

# Affichage des valeurs propres et des vecteurs propres
print("Valeurs propres :")
```

```
## [1] "Valeurs propres :"
```

```
print(val_prop)
```

```
## [1] 1.2925768 1.1269836 0.8721351 0.7083045
```

```
print("Vecteurs propres :")
```

```
## [1] "Vecteurs propres :"
```

```
print(data.frame(vec_prop))
```

```
##           X1         X2         X3          X4
## 1 -0.4935320 -0.4719478  0.5853169  0.43714484
## 2  0.4169977 -0.6547607  0.2544298 -0.57677272
## 3  0.6773621 -0.2243550 -0.1246546  0.68942484
## 4  0.3517380  0.5460939  0.7596913 -0.03051248
```

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 :

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

En prenant le produit scalaire des deux équations, nous obtenons :

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

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) 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.

Dans notre cas, chaque $\mathbf{X_t}$ représente un tableau de données à une période de temps spécifique, où les lignes représentent les individus et les colonnes représentent les variables. Par conséquent, chaque colonne de $\mathbf{X_t}$ représente une variable différente observée à cette période.

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

Lorsque nous obtenons les vecteurs propres u_k de la matrice des coefficients RV, nous avons montré précédemment qu'ils forment une base orthogonale au sens du produit scalaire euclidien. Cela signifie que ces vecteurs propres sont mutuellement orthogonaux.

Ensuite, les composantes principales F^k sont obtenues en projetant les tableaux X_t sur les vecteurs propres u_k. Comme chaque vecteur propre u_k est orthogonal aux autres vecteurs propres, les projections des tableaux sur ces vecteurs propres le seront également. Par conséquent, les composantes principales F^k seront orthogonales les unes aux autres.

c) Dépliage du tableau cubique en un tableau juxtaposé :

```r
# Création des tableaux X_t dépliés
A_jux <- matrix(A, ncol = 1)
B_jux <- matrix(B, ncol = 1)
C_jux <- matrix(C, ncol = 1)
D_jux <- matrix(D, ncol = 1)


mat_jux <- cbind(A_jux, B_jux, C_jux, D_jux)

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)
}

mat_sym <- oper_inert(mat_jux)

  result_propre = eigen(mat_sym)# list ayant les valeurs et vecteurs propres
  val_prop_jux = result_propre$values #valeurs propres
  vec_prop_jux = result_propre$vectors #vecteurs propres associées
```

La fonction **oper_inert()** est définie pour calculer une matrice de covariance normalisée: Cette fonction prend une matrice **X** en entrée, calcule une matrice de poids **W**, puis calcule une matrice de normalisation **M**. Enfin, elle renvoie la matrice de covariance normalisée **mat_d**.

Composantes principales des tableaux dépliés :

```
# On veut donner la fonction des composantes principales correspondant aux vecteurx 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 kième 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]
```

```r
  f_k = F_k / sqrt(sum(F_k^2)) # calcul de la kième 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 :
```

```r
print(head(pc1))
```

```
## [1] -1.3350171  2.6149460 -0.6205920  0.5367399  0.4548871  0.1119155
```

```r
cat("\nExtrait de la 1ère composante normée :\n")
```

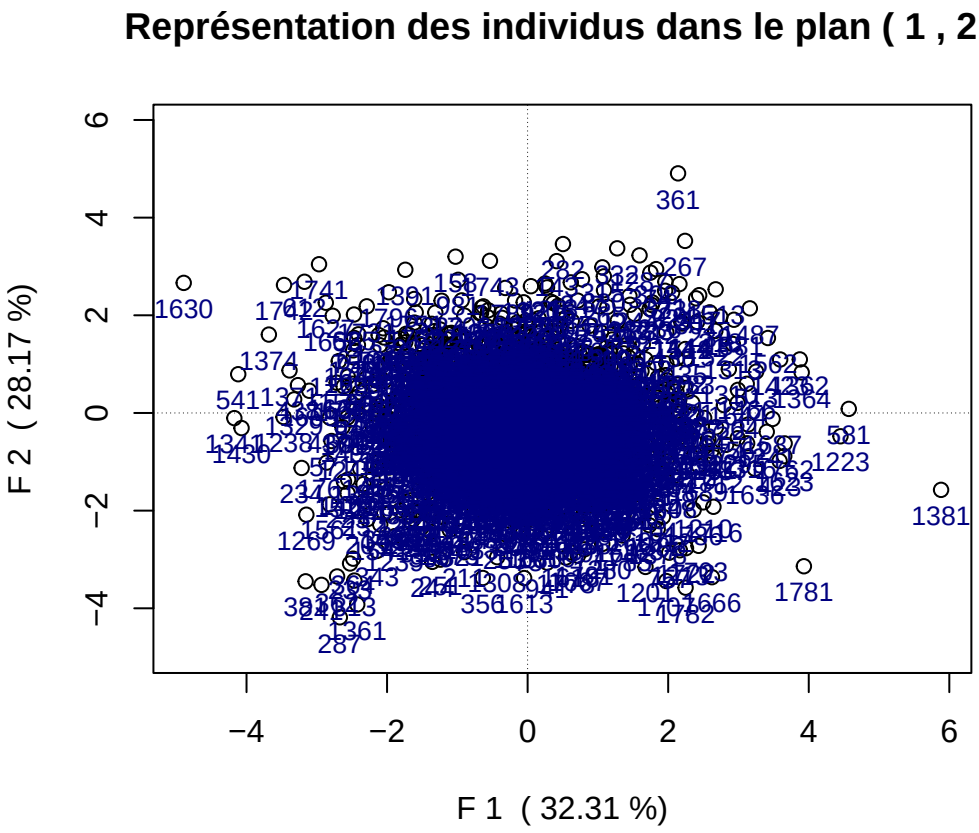
```
##
## Extrait de la 1ère composante normée :
```

```r
print(head(pc1_nr))
```

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

```r
# 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)
```



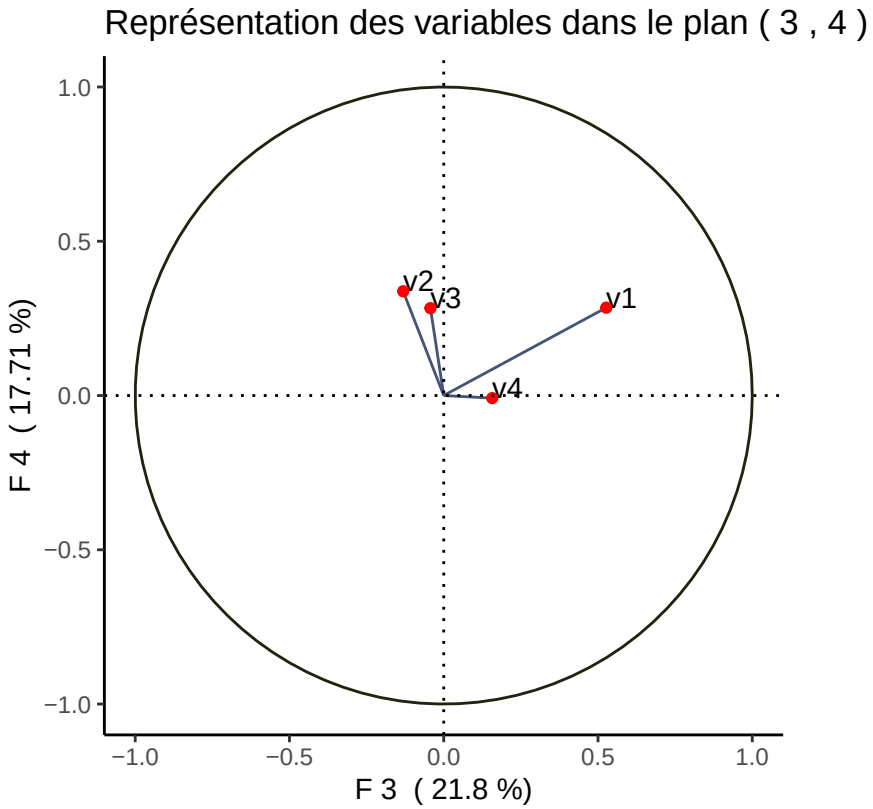
d) Cosinus entre les tableaux et les composantes principales :

```
# 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^2)))

    }

  }
  rownames(mat_cos) <- paste("v", 1:ncol(mat_cos), sep = "")
  return(mat_cos)
}
```

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

```
##           [,1]       [,2]        [,3]        [,4]
## v1 -0.3189640  0.2695008  0.43777127  0.22732417
## v2 -0.2659387 -0.3689523 -0.12642221  0.30771944
## v3  0.2552377  0.1109486 -0.05435783  0.33127671
## v4  0.1548158 -0.2042654  0.24416137 -0.01080606
```

```r
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)
```



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:

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

En utilisant le produit scalaire défini par:

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

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:

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

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

Pour $[A \mid C]$:

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

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

Pour $[B \mid C]$:

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

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

$[|A|] = \sqrt{\frac{11}{6}}$, $[|B|] = \sqrt{\frac{7}{6}}$ 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 :

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

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 $(ici \quad p = 3, \quad n = 6)$ dont le coefficient $M_{ii} = \frac{1}{||X_i||_W}$ avec:

- $W := \frac{1}{n} I_n$
- X_i la ième 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.

```r
# 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])
  }
```

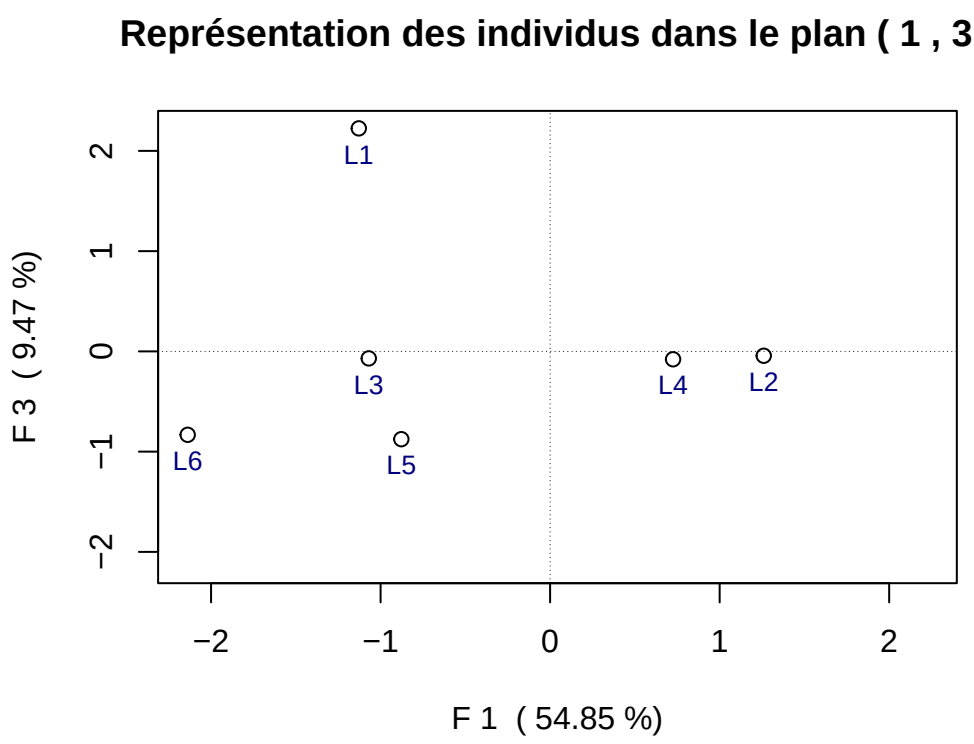
```
  return(M %*% t(X) %*% W %*% X %*% M)
}

# 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
```

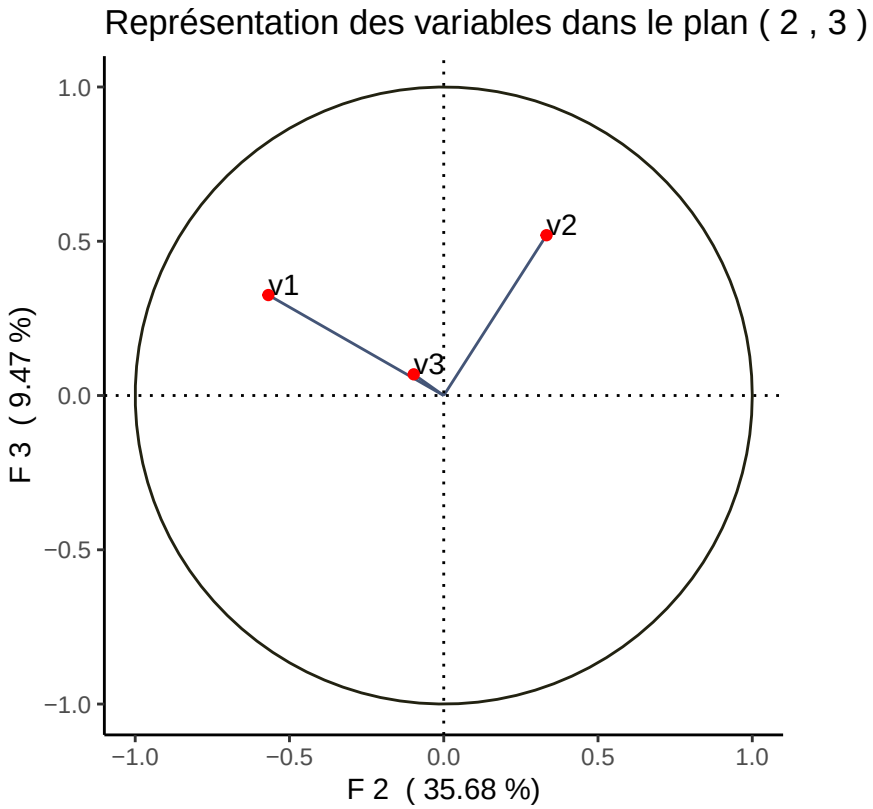
```
result_propre = eigen(Delta)# list ayant les valeurs et vecteurs propres
valp = result_propre$values #valeurs propres
vecp = result_propre$vectors
```

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)
```




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



1.2.4 Quelles ACP d'autres "dépliages" 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 définit 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 x n contenant $\frac{n(n+1)}{2}$ éléments indépendants.

1.2.a) Les poids naturel des lignes de P_m serait $W = frac1nI_n$ même chose pour les colonnes.

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

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

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 (correlées 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 anticorrelées. Enfin, si le coefficient RV est proche de 0, on dira que X_m et X_k sont décorrelées (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,..., 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)
```

```r
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)
```

```r
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 :

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

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 pra 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 principa

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
```