TP1 Quelques manipulations élémentaires autour de l'inertie (des vins de Loire)

Roméo Bex Tristan Rivaldi

02/10/2023

Table des matières :

Partie I

Introduction

Nous nous intéressons dans ce TP à l'étude de différents vins du département de la Loire. Nous disposons d'un jeu de données constitué de 21 vins décrits selon 31 attributs différents. 29 de ces attributs sont quantitatifs, les deux restants sont qualitatifs. Nous traduisons les variables quantitatives dans l'espace euclidien $\mathbb{R}^{29}$.

Pour cela, nous introduirons les bibliothèques suivantes :

```
import numpy as np
import pandas as pd
```

Définitions de fonctions générales

Nous aurons besoin d'utiliser des notions statistiques telles que l'inertie et le barycentre. Nous allons donc définir des fonctions qui nous permettront de les calculer plus tard.

```
def scale (X):
    means = np.mean(X, axis = 0) # vecteur dont les composantes sont les moyennes de chaque variables quantitatives
    stds = np.std(X, axis = 0) # vecteur dont les composantes sont les stds de chaque variables quantitatives
    return (X - means) / stds # centrage et reduction

def barycentre(X, w): # ici, X sera une matrice (N,J) et w est le vecteur de poids des individus .
    w_ = w.reshape((w.shape[0], 1)) # transforme le vect w en un vecteur colonne
    return np.sum(w_ * X, axis = 0) # multiplication élément par élément

def inertie(X, w):
    B = barycentre(X, w)
    w_ = w.reshape((w.shape[0], 1))
    return np.sum(w_ * ((X - B) ** 2))
#calcul des distances de chaque individu
# par rapport au barycentre
# puis somme sur les lignes
```

Chargement des données

```
df = pd.read_csv('wine.csv', header = 0, index_col = 0)
df.sample(5)
```

Centrage réduction des variables quantitatives

Dans un premier temps, nous allons extraire les variables quantitatives du jeu de données à l'aide de .iloc et nous les introduirons dans une matrice. Pour cela nous utilisons .values qui ignore les labels des axes.

```
# on centre et reduit
X = scale(X)
df_quan_scaled = pd.DataFrame(X, columns = df_quan.columns, index = df_quan.index)
df_quan_scaled.sample(5)
```

```
# récupération des variables quantitatives
df_quan = df.iloc[:, 2:]
# récupération de la matrice du nuage
X = df_quan.values
```

La transformation suivante nous permet de centrer et réduire chaque variable quantitative :

$$\frac{x_i^j - \overline{x}^j}{\sigma^j}.$$

Le centrage-réduction des variables quantitatives se fera grâce à la fonction scale crée plus haut.

```
# on centre et reduit
X = scale(X)
df_quan_scaled = pd.DataFrame(X, columns = df_quan.columns, index = df_quan.index)
df_quan_scaled.sample(5)
```

	Odor.Intensity.before.shaking	Aroma.quality.before.shaking	Fruity.before.shaking	Flower.before.shaking	Spice.before.shaking	Visual.inten
2ING	-1.657279	-1.294384	-0.907574	-1.198850	-0.784468	-2.518
PER1	0.997806	0.663921	0.280112	-0.135584	-0.305603	1.410
1VAU	-1.073160	-2.256096	-1.514160	-1.027586	0.720538	-1.985
2EL	-0.131490	-0.228030	-0.000243	1.591331	-0.134579	0.677
2BOU	-0.899694	0.325079	-0.698582	-0.121312	0.365664	0.944

5 rows × 29 columns

Barycentre à l'origine

Posons $\mathbf{N = 21}$ Le nombre d'observations et **J=29** le nombre de variables quantitatives, $\mathbf{z_i^j}$ la variable centrée réduite. Nous avons centré et réduit les variables quantitatives. Ainsi $\forall (i,j) \in \{1,...,N\} \times \{1,...,J\}$, $\mathbf{z_i^j} = \dfrac{x_i^j - \overline{x}^j}{\sigma^j}$

avec $\overline{x}^j$ et σ^j la moyenne et l'écart-type de la variable quantitative j.

Le barycentre du nuage est un vecteur $T \in \mathbb{R}^{29}$ tel que :

$$\mathbf{T} = (\textstyle\sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{w_i z_i^1}, \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{w_i z_i^2}, ..., \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{w_i z_i^J})$$

Comme les poids sont identiques, ceci revient donc à calculer la moyenne de chaque variable centrée-réduite et donc $w_i = \frac{1}{N}$ $\forall i$

Ainsi on a :

$$\begin{aligned}\sum_{i=1}^{N} w_i z_i^j &= \sum_{i=1}^{N} \frac{1}{N} z_i^j \\ &= \sum_{i=1}^{N} \frac{1}{N} \frac{x_i^j - \overline{x}^j}{\sigma^j} \\ &= \frac{1}{\sigma^j} \left(\sum_{i=1}^{N} \frac{1}{N} x_i^j - \sum_{i=1}^{N} \frac{1}{N} \overline{x}^j \right) \\ &= \frac{1}{\sigma^j} \left(\sum_{i=1}^{N} \frac{1}{N} x_i^j - \sum_{i=1}^{N} \frac{1}{N} \overline{x}^j \right) = 0 \quad \text{pour tout } j \in \{1, \ldots, 29\}.\end{aligned}$$

```
w = np.repeat(1 / 21., 21.)
T = barycentre(X, w)
T.round(2)
```

```
array([-0., -0., -0.,  0., -0., -0., -0.,  0.,  0.,  0.,  0., -0.,  0.,
        0., -0.,  0.,  0.,  0.,  0.,  0.,  0., -0., -0.,  0., -0., -0.,
        0., -0., -0.])
```

Ainsi le barycentre se trouve bien à l'origine théoriquement et informatiquement.

L'inertie du nuage

Après avoir centré et réduit les variables quantitatives, nous nous intéressons maintenant à l'inertie du nuage par rapport à son barycentre situé à l'origine.

Soient $\bar{z}$ le barycentre, σ_j l'écart-type de la variable j,

On a :

$$
\begin{aligned}
I_{\bar{z}}(z_i, w_i) &= \sum_{i=1}^{N} w_i \|z_i - \bar{z}\|^2 \\
&= \sum_{i=1}^{N} \frac{1}{N} \|z_i - 0\|^2 \\
&= \sum_{i=1}^{N} \frac{1}{N} \|z_i\|^2 \\
&= \sum_{i=1}^{N} \frac{1}{N} \sum_{j=1}^{J} z_{ji}^2 \\
&= \sum_{j=1}^{J} \frac{1}{N} \sum_{i=1}^{N} z_{ji}^2 \\
&= \sum_{j=1}^{J} \sigma_j^2 \\
&= \sum_{j=1}^{J} 1 \\
&= J \\
&= 29
\end{aligned}
$$

```
w = np.repeat(1 / 21., 21.)
inertie_nuage = inertie(X, w)
inertie_nuage

[6]  ✓ 0.0s
...  29.0
```

Donc on a bien l'inertie qui est égale au nombre de variables quantitatives.

Calcul de poids

Le jeu de données dont nous disposons contient trois appellations de vins : Saumur, Chinon, Bourgueil. Ces appellations partitionnent notre dataset. Nous nous intéressons aux éventuels liens entre l'apellation et les attributs des vins de Loire.

Nous allons dans un premier temps extraire les matrices propres à chaque appellation et calculer leur poids respectifs.

```python
# extraction de trois matrices correspondantes à chaque appellation
df_bourgueuil = df_quan_scaled[df['Label'] == 'Bourgueuil']
df_chinon = df_quan_scaled[df['Label'] == 'Chinon']
df_saumur = df_quan_scaled[df['Label'] == 'Saumur']
```

```python
# calcul des poids
w_bourgueuil = df_bourgueuil.shape[0] / df.shape[0] # nb de bourgueuil / nb de lignes total
print("Poids de l'appelation Bourgueuil:", w_bourgueuil)
w_chinon = df_chinon.shape[0] / df.shape[0] #nb de chinon / nb de lignes tot.
print("Poids de l'appelation Chinon:", w_chinon)
w_saumur = df_saumur.shape[0] / df.shape[0]
print("Poids de l'appelation Saumur:", w_saumur)
```

```
Poids de l'appelation Bourgueuil: 0.2857142857142857
Poids de l'appelation Chinon: 0.19047619047619047
Poids de l'appelation Saumur: 0.5238095238095238
```

Comme ces appelations partitionnent notre data set, leur poids respectifs somment à 1.

```python
print(w_bourgueuil+w_chinon+w_saumur)
```

✓ 0.0s

```
1.0
```

Calcul des barycentres

Nous disposons maintenant des poids et des matrices de chaque appellations. Nous pouvons donc faire appel à la fonction barycentre qui nous permettra de calculer les barycentres de chaque nuage.

```python
# calcul des barycentres
X_bourgueuil = df_bourgueuil.values ## On recupère les matrices coresspondantes
w_bourgueuil_ = np.repeat(1. / df_bourgueuil.shape[0], df_bourgueuil.shape[0]) #poids matrice bourgeuil
B_bourgueuil = barycentre(X_bourgueuil, w_bourgueuil_)
print("Le barycentre de Bourgueuil:\n", B_bourgueuil.round(2))
X_chinon = df_chinon.values
w_chinon_ = np.repeat(1. / df_chinon.shape[0], df_chinon.shape[0])
B_chinon = barycentre(X_chinon, w_chinon_)
print("Le barycentre de Chinon:\n", B_chinon.round(2))
X_saumur = df_saumur.values
w_saumur_ = np.repeat(1. / df_saumur.shape[0], df_saumur.shape[0])
B_saumur = barycentre(X_saumur, w_saumur_)
print("Le barycentre de Saumur:\n", B_saumur.round(2))
```

```
Le barycentre de Bourgueuil:
 [-0.63  0.08  0.04 -0.18  0.04 -0.32 -0.3  -0.23 -0.84 -0.1  -0.01  0.15
  0.09 -0.75 -0.91 -0.39 -0.29  0.28 -0.16 -0.15 -0.15 -0.29  0.19 -0.
 -0.47 -0.05 -0.03  0.26 -0.03]
Le barycentre de Chinon:
 [-0.6  -0.83  0.19 -0.29 -0.2  -0.67 -0.61 -0.32 -0.32  0.16  0.17 -0.1
 -0.33  0.37  0.02 -0.33 -0.25 -0.1  -0.41  0.95 -0.69  0.04 -0.43 -0.48
 -0.08 -0.48 -0.55 -0.39 -0.19]
Le barycentre de Saumur:
 [ 0.56  0.26 -0.09  0.2   0.05  0.42  0.39  0.24  0.57 -0.   -0.06 -0.04
  0.07  0.27  0.49  0.33  0.25 -0.12  0.24 -0.27  0.33  0.14  0.06  0.18
  0.29  0.2   0.21 -0.    0.08]
```

Normes euclidiennes des trois barycentres

```
ne_bourgueuil = np.sqrt(np.sum(B_bourgueuil ** 2))**2
print("La norme euclidienne du barycentre Bourgueuil:", ne_bourgueuil)
ne_chinon = (np.sum(B_chinon ** 2))
print("La norme euclidienne du barycentre Bourgueuil:", ne_chinon)
ne_saumur = (np.sum(B_saumur ** 2))
print("La norme euclidienne du barycentre Saumur:", ne_saumur)
```

```
La norme euclidienne du barycentre Bourgueuil: 3.60352706943803
La norme euclidienne du barycentre Bourgueuil: 5.399969967209555
La norme euclidienne du barycentre Saumur: 2.1197784244208946
```

Calcul de l'inertie inter-appelations

Une fois que nous avons défini les barycentres de chaque appellation, nous nous intéressons aux disparités entre classes.

$$\sigma^2_{\mathbf{inter}} = \sum_{j=1}^{3} w_j \|z_j - \bar{z}\|^2$$

```
iia = w_bourgueuil * ne_bourgueuil + w_chinon * ne_chinon + w_saumur * ne_saumur
print("L'intertie inter-appellations est:", iia)
```

```
L'intertie inter-appellations est: 3.1685049978141064
```

Calcul du R^2 :

L'indicateur de liaison R^2 ous donne de l'information sur l'influence d'une classe sur la dispersion des nuages entre eux.

$$R^2 = \frac{\sigma^2_{\mathbf{inter}}}{\sigma^2_{\mathbf{tot}}}$$

Ainsi avec un R^2 proche de 1 et grâce au Théorème de Huygens, on récupère de l'information sur la variance intra-classe qui nous indique que les éléments de la même classe sont assez rapprochés.

```
r2 = iia / inertie_nuage
r2.round(2)
```

```
0.11
```

Le R^2 de la partition des vins en appellations vaut 0.11. Donc, l'appellation n'explique qu'environ 11% des disparités sensorielles entre les vins de Loire.

Relation entre l'appellation et les autres variables

Nous avons étudié l'influence de l'appellation sur les disparités par rapport à l'ensemble des variables du jeu de données.

Nous nous intéressons maintenant à l'étude de l'influence de l'apellation sur chaque variable sensorielle. Nous allons calculer, dans un premier temps, les moyennes intérieures de chaque classe (Saumur,Bourgueuil, Chinon) pour chaque variable. Si j est une classe, alors la moyenne à l'intérieur de cette classe par rapport à la variable x est :

$$\bar{x}_j = \sum_{i \in \mathbf{cl}_j} \frac{w_i}{w_j} x_i$$

avec $w^j = \sum_{i \in \mathrm{cl}_j} w_i$ et $(x_i)_{i \in [1,N]}$ les coordonnées de la variable x

Par exemple pour la classe Bourgueil, et pour une variable donnée :

$$\bar{x}_{bourg} = \sum_{i=1}^{6} \frac{\frac{1}{6}}{\frac{1}{6}}$$

Nous faisons la même chose pour les classes Chinon et Saumur et nous procédons au calcul de la moyenne globale :

$$\bar{x} = \sum_{j=1}^{3} w^j \bar{x}^j$$

```
x_bar_bourgueuil = ((1. / df_quan.shape[0]) / w_bourgueuil) * df_bourgueuil.sum(axis = 0)
x_bar_chinon = ((1. / df_quan.shape[0]) / w_chinon) * df_chinon.sum(axis = 0)
x_bar_saumur = ((1. / df_quan.shape[0]) / w_saumur) * df_saumur.sum(axis = 0)

moyennes_globales = w_bourgueuil * x_bar_bourgueuil + w_chinon * x_bar_chinon + w_saumur * x_bar_saumur
```

Une fois les moyennes à disposition, nous nous intéressons à la variance inter-classe qui représente la dispersion des moyennes des classes autour de la moyenne globale. Il nous faut une référence à laquelle comparer la variance inter-classe. C'est pour cela que nous nous intéressons aussi à la variance totale et notamment au rapport entre les deux.

On a :

$$\sigma^2_{inter} = \sum_{j=1}^{3} w^j (\bar{x}^j - \bar{x})^2$$
$$\sigma^2_{tot} = \sum_{j=1}^{3} \sum_{i \in cl.j} w_i (x_i - \bar{x})^2$$

Or, les variables sont centrées-réduites, leurs variances valent 1 et ainsi on a :

$$R^2 = \sigma^2_{inter}$$

```
variances_inter_classes = w_bourgueuil * ((x_bar_bourgueuil - moyennes_globales) ** 2) \
                        + w_chinon * ((x_bar_chinon - moyennes_globales) ** 2) \
                        + w_saumur * ((x_bar_saumur - moyennes_globales) ** 2)

variances_totales = np.repeat(1.,29)

r2s = variances_inter_classes
df_rel = pd.DataFrame({
    'variance inter-classes': variances_inter_classes,
    'variance totale': variances_totales,
    'r2': r2s
})
df_rel.sort_values('r2', ascending = False)
```

	variance inter-classes	variance totale	r2
Odor.Intensity	0.390584	1.0	0.390584
Phenolic	0.360217	1.0	0.360217
Odor.Intensity.before.shaking	0.342045	1.0	0.342045
Plante	0.223474	1.0	0.223474
Acidity	0.214199	1.0	0.214199
Visual.intensity	0.206716	1.0	0.206716
Nuance	0.174013	1.0	0.174013
Aroma.quality.before.shaking	0.169997	1.0	0.169997
Astringency	0.154795	1.0	0.154795
Aroma.intensity	0.123684	1.0	0.123684
Bitterness	0.107271	1.0	0.107271
Harmony	0.081194	1.0	0.081194
Aroma.persistency	0.069346	1.0	0.069346
Attack.intensity	0.068403	1.0	0.068403
Surface.feeling	0.065967	1.0	0.065967
Intensity	0.064985	1.0	0.064985
Smooth	0.059088	1.0	0.059088
Overall.quality	0.049219	1.0	0.049219
Balance	0.047724	1.0	0.047724
Flavor.before.shaking	0.046910	1.0	0.046910

Les variables quantitatives ci-dessus sont classées par leurs degrés de liaison (explicative) à l'appellation (à la variable Label). Ainsi, Odor.Intensity explique environ 40% de la disparité des classes. De même pour les autres variables.

```
df_rel['r2'].sum()/29
```

```
0.1092587930280726
```

Ainsi le R^2 de la partition est égal à la moyenne arithmétique des R^2 des variables.

Partie II

1) Projecteurs et interprétation statistique

Rappelons la définition d'un W-projecteur sur un sous-espace vectoriel :

$$\Pi_Y = Y(Y'WY)^{-1}Y'W$$

Rappelons aussi que pour la matrice Y^c dont les colonnes sont les colonnes centrées de Y, on a :

$$\Pi_Y = \Pi_{Y^c} + \Pi_{\mathbb{1}}$$

La matrice X est centrée réduite donc chaqu'une de ses colonnes x^j possède une moyenne nulle et : $\forall\ 1 \leq j \leq p : \Pi_{\mathbb{1}}\ x^j = \overline{x^j}\ \mathbb{1} = \mathbb{0}$

On obtient donc: $\forall\ 1 \leq j \leq p : \Pi_Y\ x^j = \Pi_{Y^c}\ x^j$

On va maintenant calculer puis interpréter statistiquement $\|\Pi_Y\ x^j\|_W^2$:

$$\begin{aligned}
\|\Pi_Y x^j\|_W^2 &= \langle \Pi_Y x^j | \Pi_Y x^j \rangle_W \quad \text{car les projecteurs sont auto-adjoints} \\
&= \langle x^j | \Pi_Y x^j \rangle_W \\
&= \|x^j\| \|\Pi_Y x^j\| cos(x^j, \Pi_Y x^j) \\
&= \|\Pi_Y x^j\| cos(x^j, \Pi_Y x^j) \quad \text{car } x^j \text{ est réduite}
\end{aligned}$$

On en déduit alors que :

$$\|\Pi_Y x^j\|_W = cos(x^j, \Pi_Y x^j)$$

Donc en élevant au carré on obtient :

$$\begin{aligned}
\|\Pi_Y x^j\|_W^2 &= (cos(x^j, \Pi_Y x^j))^2 \\
&= (cos(x^j, \Pi_{Y^c} x^j))^2
\end{aligned}$$

Or x^j est centré réduite donc sa variance vaut 1 (i.e $\|x^j\|_W^2$=1) donc:

$$\|\Pi_Y x^j\|_W^2 = R^2$$

Et on a que $\|\Pi_Y x^j\|_W^2$ correspond à la variance inter-classe R^2 de x^j.

2) Calcul de la trace

Nous programmons Π_Y puis Π_{x^j} et calculons tr(Π_{x^j},Π_Y).On obtient:

$$tr(\Pi_{x^j}, \Pi_Y) = tr(x^j(x^{j'}Wx^j)^{-1}(x^{j'}W\Pi_Y))$$

Or $x^{j'}Wx^j = \|x^j\|^2_W$ et $\|x^j\|^2_W$=1 donc :

$$tr(x^j(x^{j'}Wx^j)^{-1}(x^{j'}W\Pi_Y)) = tr(x^j x^{j'}W\Pi_Y)$$

Par invariance de la trace par permutation circulaire on obtient:

$$tr(x^j(x^{j'}Wx^j)^{-1}(x^{j'}W\Pi_Y)) = tr(x^{j'}W\Pi_Y x^j)$$

et comme $x^{j'}W\Pi_Y x^j$ est un scalaire :

$$tr(x^{j'}W\Pi_Y x^j) = x^{j'}W\Pi_Y x^j = \langle\, x^j|\Pi_Y x^j\rangle_W$$

Comme les projecteurs sont idempotents on a:

$$\langle\, x^j|\Pi_Y x^j\rangle_W = \langle\, x^j|(\Pi_Y)^2 x^j\rangle_W$$

Et:

$$\langle\, x^j|(\Pi_Y)^2 x^j\rangle_W = \langle\, \Pi_Y x^j|\Pi_Y x^j\rangle_W$$

On peut donc conclure que:

$$tr(\Pi_{x^j}, \Pi_Y) = \|\Pi_Y x^j\|^2_W$$

Donc d'aprées la question précédente tr(Π_{x^j}, Π_Y) correspond au R^2 de x^j si l'on partitionne selon les appellations. Voici un extrait des valeurs calculées pour j allant de 1 à 5.

j	$tr(\Pi_{x^j}, \Pi_Y)$
1	0.342045343
2	0.169997137
3	0.011456314
4	0.046909534
5	0.009229911

3) Calcul de la tr(RΠ_Y) avec R=XMX'W

Nous savons que le R^2 de la partition est égale à la moyenne arthmétique des R^2 des variables. On a donc:

$$
\begin{aligned}
R^2 &= \frac{1}{29}\sum_{j=1}^{29} \|\Pi_Y x^j\|_W^2 \\
&= \frac{1}{29}\sum_{j=1}^{29} \langle\, \Pi_Y x^j | \Pi_Y x^j \rangle_W \\
&= \frac{1}{29}\sum_{j=1}^{29} \langle\, x^j | (\Pi_Y)^2 x^j \rangle_W \quad \text{car les projecteurs sont auto-adjoints} \\
&= \frac{1}{29}\sum_{j=1}^{29} \langle\, x^j | \Pi_Y x^j \rangle_W \quad \text{car les projecteurs sont idempotents} \\
&= \frac{1}{29}\sum_{j=1}^{29} x^{j'} W \Pi_Y x^j \\
&= \frac{1}{29}\sum_{j=1}^{29} m^j x^{j'} W \Pi_Y x^j \\
&= \frac{1}{29}\sum_{j=1}^{29} (MX'W\Pi_Y X)_{j,j} \\
&= tr(MX'W\Pi_Y X) \\
&= tr(XMX'W\Pi_Y X) \\
&= tr(R\Pi_Y)
\end{aligned}
$$

Nous avons alors $tr(R\Pi_Y)$=R^2 total si nous partitionnons selon les appelations. On a obtnue R^2= 0,11 arrondie à 10^{-2} prés. On remarque que l'on retrouve bien le meme résultat que dans la première partie du TP.

4) Calcul et interprétation statistique de $tr(R\Pi_Z)$ et de $tr(\Pi_{x^j}\Pi_Z)$

Le calcul de $tr(\Pi_{x^j}\Pi_Z)$ ce fait de la meme manière que celui de $tr(\Pi_{x^j}\Pi_Y)$ effectué dans la partie précédente. On a donc que $tr(\Pi_{x^j}\Pi_Z)$=$\|\Pi_Z x^j\|_W^2$ et $tr(\Pi_{x^j}\Pi_Z)$ correspond au R^2 de x^j si nous partionnons selon les sols. Voici un extrait des valeurs calculées pour j allant de 1 à 5.

j	$tr(\Pi_{x^j}, \Pi_Z)$
1	0.5341064
2	0.4152065
3	0.3231181
4	0.3466322
5	0.5576593

Nous avons aussi que $tr(R\Pi_Z)$ est égal au R^2 total de la partition induite par Z. Nous obtenons R^2=0.36 arrondie à 10^{-2} prés.

5) Code R

```r
# Importation du fichier csv

setwd(path.expand("~/Desktop"))
data <- read.csv("wine.csv")
```

```r
# Création des matrices W et M

M <- diag(1/29, nrow = 29, ncol = 29)
W <- diag(1/21, nrow = 21, ncol = 21)
```

```r
# Creéation de la matrice X

standard <- function(x){(x-mean(x))/(sd(x)*sqrt(1-1/length(x)))}
X <- apply(as.matrix(data[,sapply(data,is.numeric)]), 2, standard)


# Création de la matrice Y

Label_uniques <- unique(data$Label)
Y <- model.matrix(~ Label - 1, data=data)

# Création de la matrice Z

valeurs_soil_uniques <- unique(data$Soil)
Z <- model.matrix(~ Soil - 1, data=data)
```

```r
# Création de la matrice de projection de Y

Pi_Y = Y%*%solve(t(Y)%*%W%*%Y)%*%t(Y)%*%W
```

```r
# Fonction pour calculer la projection Pxj

Pxj = function(i){ X[,i]%*%solve(t(X[,i])%*%W%*%X[,i])%*%X[,i]%*%W }
```

```r
#Trace du produit Pxj par Pi_Y

trace_of_product <- function(A, B) {
sum(diag(A %*% B))
}
trace_Pi_Y_Pxj <- function(j, Pi_Y) {
Pxj_j <- Pxj(j)
trace_of_product(Pi_Y, Pxj_j)
}
j_values <- 1:13
results_df <- data.frame(j = j_values, trace = numeric(length(j_values)))
for (i in 1:length(j_values)) {
results_df$trace[i] <- trace_Pi_Y_Pxj(j_values[i], Pi_Y)
}
print(results_df)
```

```r
# Création de la matrice R

R <- X %*% M %*% t(X) %*% W
```

```r
# Calcule de la trace de R * Pi_Y

trace_result <- sum(diag(R %*% Pi_Y))
print(trace_result)
```

```r
# Création de la matrice de projection de Z

Pi_Z = Z%*%solve(t(Z)%*%W%*%Z)%*%t(Z)%*%W
```

```r
#Trace du produit Pxj par Pi_Z

trace_of_product <- function(A, B) {
sum(diag(A %*% B))
}
trace_Pi_Z_Pxj <- function(j, Pi_Z) {
Pxj_j <- Pxj(j)
trace_of_product(Pi_Z, Pxj_j)
}
j_values <- 1:13
results_df <- data.frame(j = j_values, trace = numeric(length(j_values)))
for (i in 1:length(j_values)) {
results_df$trace[i] <- trace_Pi_Z_Pxj(j_values[i], Pi_Z)
}
print(results_df)
```

```r
# Calcule de la trace de R * Pi_Z

trace_result <- sum(diag(R %*% Pi_Z))
print(trace_result)
```