
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
École Nationale Supérieure d'Informatique (ESI ex. INI)



Analyse de Données

Rapport du TP

Application de l'ACP et interprétation des résultats

Réalisé par:

- **Fentazi Mohamed Readh**

Groupe:

- **SID 1**

1 Introduction

L'analyse des composantes principales (ACP) permet de transformer des variables corrélées en variables décorrélées baptisée "composantes principales". Plus précisément, cette méthode vise à réduire le nombre de variables appliquées à des individus, pour simplifier les observations tout en conservant un maximum d'informations. Seules une, deux ou trois variables dites "composantes principales" sont conservées.

Dans le cadre de ce TP, nous allons l'utiliser pour interpréter les données issus des notes l'an dernier.

2 Librairies utilisés:

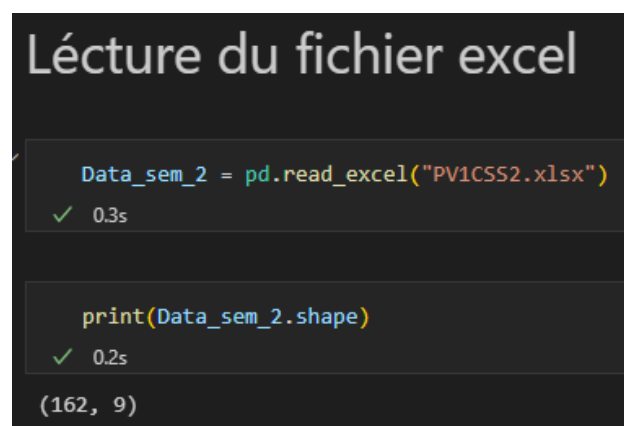
```
import pandas as pd # Lecture des fichiers excel et prétraitement
import numpy as np # Calcule etc...
from sklearn.decomposition import PCA # Pour faire l'ACP
from sklearn import preprocessing # Pour centrer et réduire nos donnée
import matplotlib.pyplot as plt # Pour dessiner les graphes
```

3 Lecture des données

Nous avons utilisé `pd.read_excel` pour lire les données représentant les notes du S2 du fichier Excel *PV1CSS1.xlsx*

Remarque:

(Le choix du deuxième semestre est subjective et je l'ai choisi car c'est le semestre où j'ai eu de meilleures notes généralement, classé 32ème de la promo comparé au 1er semestre où je suis classé 68ème de la promo espérant être bien représenté 😊)



```
Lecture du fichier excel

Data_sem_2 = pd.read_excel("PV1CSS2.xlsx")
✓ 0.3s

print(Data_sem_2.shape)
✓ 0.2s

(162, 9)
```

- En affichant quelques lignes de nos données (les 5 premières et 5 dernières) comme suit:

Data_sem_2
✓ 0.6s

	MCSI * x 5	BDD * x 5	SEC * x 1	CPROJ * x 3	PROJ * x 3	LANG2 * x 2	ARCH * x 4	SYS2 * x 4	RES2 * x 3
0	14.66	17.60	17.12	15.15	16.89	17.34	17.59	18.90	16.34
1	12.56	17.55	17.42	15.55	15.71	16.93	17.66	16.90	17.62
2	10.69	15.70	16.21	16.33	15.71	17.10	16.90	16.10	15.98
3	12.53	15.66	15.34	15.08	16.55	16.92	16.05	14.20	16.70
4	13.76	16.35	9.62	14.05	16.24	16.09	14.60	15.10	16.15
...
157	MCSI: 5.56 < 6.26	8.85	SEC: 3.58 < 5.00	9.45	11.10	14.77	12.29	SYS2: 5.72 < 6.21	9.59
158	MCSI: 5.79 < 6.26	BDD: 6.69 < 7.33	SEC: 1.79 < 5.00	11.85	16.00	13.02	10.05	SYS2: 2.80 < 6.21	7.72
159	MCSI: 2.68 < 6.26	8.65	SEC: 1.58 < 5.00	13.30	14.40	10.30	9.44	SYS2: 4.30 < 6.21	9.46
160	11.52	BDD: 3.10 < 7.33	SEC: 1.75 < 5.00	15.38	16.99	12.56	ARCH: 4.61 < 7.81	SYS2: 1.20 < 6.21	RES2: 2.34 < 7.53
161	MCSI: 0.00 < 6.26	BDD: 0.00 < 7.33	SEC: 0.00 < 5.00	CPROJ: 0.00 < 8.01	PROJ: 0.00 < 9.13	LANG2: 0.00 < 8.55	ARCH: 0.00 < 7.81	SYS2: 0.60 <	NaN

162 rows × 9 columns

Nous remarquons que :

1. Les noms des colonnes représentant les modules contiennent les coefficients de ces derniers (On suppose que ces données sont inutiles pour notre étude).
2. Quelques lignes dans notre ensemble de données contiennent des données manquantes.
3. Certains données sont incohérentes, on remarque que les moyennes en dessous de la note minimale de chaque module sont représentées comme suit :
nom du module : moyenne de X en **ce module** < note minimale de **ce module**.
4. Le type des données n'est pas numérique/alphabétique (on verra ça sur le notebook).

4 Prétraitement des données

4.1 Renommage des colonnes

La méthode rename de dataframe pd a été utilisée pour renommer nos colonnes de cette façon :

```
data_renomme = Data_sem_2.rename(columns={"MCSI\xa0*\xa0x 5": "MCSI", "BDD\xa0*\xa0x 5": "BDD", ...
```

4.2 Suppression des lignes inutiles

Les dernières lignes de notre tableau de données contenaient des données manquantes

```
Data_non_nul = data_renomme.dropna(subset=["MCSI", "BDD", "SEC", "CPROJ", "PROJ", "LANG2", "ARCH", "SYS2", "RES2"])
```

4.3 Correction des incohérences

Nous devons changer ces cellules “moyenne module < note éliminatoire du module” en moyenne seulement comme les autres cellules

```
Data_coherente.MCSI = Data_coherente['MCSI'].mask(Data_coherente['MCSI'].str.len() > 8, Data_coherente['MCSI'].str[-5:])
Data_coherente.BDD = Data_coherente['BDD'].mask(Data_coherente['BDD'].str.len() > 8, Data_coherente['BDD'].str[-5:])
Data_coherente.SEC = Data_coherente['SEC'].mask(Data_coherente['SEC'].str.len() > 8, Data_coherente['SEC'].str[-5:])
Data_coherente.CPROJ = Data_coherente['CPROJ'].mask(Data_coherente['CPROJ'].str.len() > 8, Data_coherente['CPROJ'].str[-5:])
Data_coherente.PROJ = Data_coherente['PROJ'].mask(Data_coherente['PROJ'].str.len() > 8, Data_coherente['PROJ'].str[-5:])
Data_coherente.LANG2 = Data_coherente['LANG2'].mask(Data_coherente['LANG2'].str.len() > 8, Data_coherente['LANG2'].str[-5:])
Data_coherente.ARCH = Data_coherente['ARCH'].mask(Data_coherente['ARCH'].str.len() > 8, Data_coherente['ARCH'].str[-5:])
Data_coherente.SYS2 = Data_coherente['SYS2'].mask(Data_coherente['SYS2'].str.len() > 8, Data_coherente['SYS2'].str[-5:])
Data_coherente.RES2 = Data_coherente['RES2'].mask(Data_coherente['RES2'].str.len() > 8, Data_coherente['RES2'].str[-5:])
```

4.4 Correction des types des variables

Nos données était de type objet et non pas numérique donc on les a changées:

```
Data_finale['MCSI'] = pd.to_numeric(Data_finale['MCSI'])
Data_finale['BDD'] = pd.to_numeric(Data_finale['BDD'])
Data_finale['SEC'] = pd.to_numeric(Data_finale['SEC'])
Data_finale['CPROJ'] = pd.to_numeric(Data_finale['CPROJ'])
Data_finale['PROJ'] = pd.to_numeric(Data_finale['PROJ'])
Data_finale['LANG2'] = pd.to_numeric(Data_finale['LANG2'])
Data_finale['ARCH'] = pd.to_numeric(Data_finale['ARCH'])
Data_finale['SYS2'] = pd.to_numeric(Data_finale['SYS2'])
Data_finale['RES2'] = pd.to_numeric(Data_finale['RES2'])
```

Résultat:

Data_finale.dtypes	
✓	0.7s
MCSI	float64
BDD	float64
SEC	float64
CPROJ	float64
PROJ	float64
LANG2	float64
ARCH	float64
SYS2	float64
RES2	float64
dtype:	object

5 Statistiques du tableau de données:

```
Data_finale.describe()
```

✓ 0.1s

	MCSI	BDD	SEC	PROJ	LANG2	ARCH	SYS2	RES2
count	161.000000	161.000000	161.000000	161.000000	161.000000	161.000000	161.000000	161.000000
mean	10.544286	12.326335	8.620000	15.318012	14.405714	13.119876	10.498137	12.667764
std	1.840143	1.930593	3.207341	1.346793	2.001129	1.922817	2.526939	1.770068
min	6.260000	7.330000	3.620000	11.100000	8.550000	7.810000	6.210000	7.530000
25%	9.340000	11.200000	5.380000	14.440000	13.750000	11.800000	8.550000	11.540000
50%	10.620000	12.150000	8.250000	15.410000	14.790000	13.100000	10.570000	12.660000
75%	11.860000	13.600000	10.790000	16.500000	15.650000	14.620000	12.300000	13.840000
max	14.660000	17.600000	17.420000	17.910000	17.510000	17.890000	18.900000	17.620000

6 L'analyse des composantes principales

6.1 Normalisation des données:

Nous devons d'abord normaliser nos données comme suit:

```
Donnes_normalise = preprocessing.scale(Data_finale.T) # On passe la transposé
```

On a maintenant ces données:

	0	1	2	3	4	5	6
0	-1.77788...	-2.48281...	-2.73823...	-2.21796...	-0.44863...	-0.97037...	-0.84028...
1	0.616152...	0.715785...	0.035681...	0.161321...	0.839258...	1.097344...	-0.54424...
2	0.225289...	0.632455...	0.318056...	-0.08192...	-2.50727...	0.491991...	-1.84145...
3	-1.37887...	-0.56621...	0.384497...	-0.27956...	-0.30443...	0.213120...	0.424629...
4	0.038000...	-0.46365...	0.041218...	0.837859...	0.784560...	1.022525...	1.307381...
5	0.404434...	0.318364...	0.810829...	1.119117...	0.709971...	0.750456...	1.188963...
6	0.608009...	0.786296...	0.700093...	0.457782...	-0.03094...	-0.32421...	1.022102...
7	1.674739...	0.299134...	0.257152...	-0.94850...	0.217687...	-2.20149...	-0.32893...
8	-0.40986...	0.760655...	0.190710...	0.951883...	0.739807...	-0.07935...	-0.38814...

Remarque:

In R using `scale()` or `prcomp()`, variation is calculated as:

$$\frac{(\text{measurements} - \text{mean})^2}{\text{the number of measurements} - 1}$$

Il y'a une petite différence entre le calcul de la variance sur R et sur SKLEARN (python) ou autre méthode utilisée. En effet, sur R en calculant la variance on divise sur le n-1 (n étant le nombre d'observations/individus) alors que sur SKLEARN ou python on divise sur N.

Cependant, cette différence n'aura pas d'effet sur notre ACP mais notre graphe finale sera malheureusement un peu impacté.

6.2 L'ACP et choix du nombre d'axes:

En fine on peut faire note ACP:

```
ACP= PCA()
ACP.fit(Donnes_normalise)
Donnes_ACP = ACP.transform(Donnes_normalise)
pd.DataFrame(Donnes_ACP)
```

✓ 0.1s

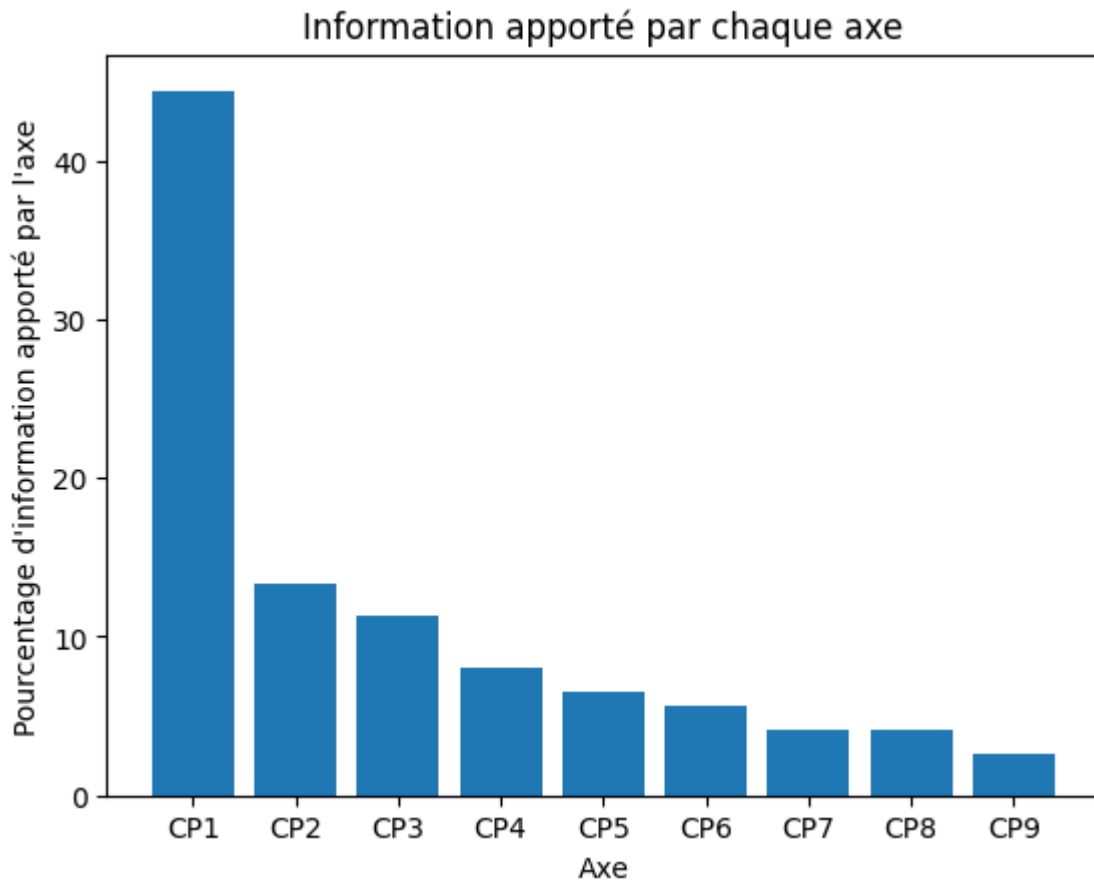
	0	1	2	3	4	5	6	7	8
0	6.629458	-0.466922	0.421273	0.244681	0.587124	-0.018480	-0.340201	0.430134	0.768978
1	6.296030	0.483786	0.119580	-0.224235	-0.070992	-0.494259	-0.367682	-0.284204	-0.171087
2	5.030829	0.453018	-0.221409	-0.359864	-1.247743	-0.885893	-0.462859	0.244817	0.079744
3	4.633791	-0.370299	-0.054340	0.380803	-0.109971	-0.591544	-0.046732	-0.316511	-0.333864
4	3.679510	-0.520736	0.438098	0.237812	0.517166	1.004558	0.425874	-0.932771	0.531286
...
156	-3.689910	3.210336	-0.500027	-0.760156	0.124401	-0.108105	0.494025	-0.079334	-0.158752
157	-4.469322	3.746753	-0.744625	0.750041	0.855722	0.311744	0.356442	0.651770	-0.581989
158	-5.042429	0.049927	0.220951	0.940001	-0.774347	-0.957250	-0.474340	0.950567	0.405339
159	-4.603523	0.291619	0.108569	-1.144752	-0.894079	-1.014139	-0.590486	-0.253726	0.055954
160	-3.962888	-3.357738	-1.269791	-0.768443	-0.847209	-0.538764	-0.125141	0.920529	0.976982

161 rows × 9 columns

Pour choisir le nombre d'axe on doit calculer le pourcentage d'information expliqué par chacun des axes: $I_i = \frac{\lambda_i}{\sum \lambda_i} \times 100$

On obtient le graphe suivant:

D'après le graphe le gain d'information est négligeable après l'ajout du 4eme axe de plus sa valeur propre est <1. Donc on retient **3 axes**.



Qualité de représentation du plan engendré par les axes 1,2,3 :

```
valeurs_propres = ACP.fit(Donnes_ACP).explained_variance_
valeurs_propres
✓ 0.8s
array([3.99930806, 1.19430853, 1.01530709, 0.72405231, 0.58664804,
       0.50593133, 0.37128893, 0.36621966, 0.23693605])
```

```
gain_info_123=valeurs_propres[0:3].sum()/valeurs_propres.sum() * 100
gain_info_123
✓ 0.4s
68.98804087497346
```

$$I_{123} = \frac{\lambda_1 + \lambda_2 + \lambda_3}{\sum \lambda_i} = 68.99\%$$

Donc **68.99%** de l'information est contenue dans le plan engendré par les axes 1, 2 et 3

7 Interprétation des axes :

Une variable X^i contribue à la construction d'un axe α , si sa contribution absolu par rapport à cette axe est supérieure à $1/p$. Avec $p = 9$

donc: **Contribution_abs > 11.11%**

Les contribution absolu de chaque variable par rapport aux 3 axes:

```
contribution_absolu = (vecteurs_propres**2 / (vecteurs_propres**2).sum(axis=0))*100
contribution_absolu_axes123 = contribution_absolu.iloc[:,0:3]
contribution_absolu_axes123
```

✓ 0.9s

	CP1	CP2	CP3
MCSI	7.256690	28.476746	4.611930
BDD	16.295296	0.716461	0.244237
SEC	13.529029	0.375849	6.626715
CPROJ	7.420873	13.858153	9.304325
PROJ	0.713613	44.153129	21.773089
LANG2	6.546158	1.612405	32.004438
ARCH	14.245715	5.586314	15.007364
SYS2	16.922205	4.462656	8.795225
RES2	17.070421	0.758288	1.632679

Le signe des variables sur l'axe α est déterminé par la projection de celle-ci sur l'axe α

1er axe :

-	+
	BDD,SEC,ARCH,SYS2,RES2

L'axe 1 est un axe à effet taille, il mesure les notes obtenues en **BDD,SEC,ARCH,SYS2,RES2**.

2ème axe :

-	+
MCSI,PROJ,CPROJ	

L'axe 2 est un axe à effet taille, il mesure les notes obtenues en **MCSI,PROJ,CPROJ** mais dans le sens contraire!

3ème axe :

-	+
LANG2	PROJ,ARCH

L'axe 3 est un axe d'opposition, il oppose les notes obtenues en LANG2 avec celle des notes obtenues en **PROJ** et **ARCH**.

Qualité de représentation du plan engendré par Les axes 1 et 2 et les axes 1,2 et 3

```
gain_info_12 = valeurs_propres[0:2].sum()/valeurs_propres.sum() * 100
gain_info_123 = valeurs_propres[0:3].sum()/valeurs_propres.sum() * 100
print("Qualité de représentation du plan engendré par les axes 1,2:",gain_info_12)
print("Qualité de représentation du plan engendré par les axes 1,2,3:",gain_info_123)
```

✓ 0.2s

Qualité de représentation du plan engendré par les axes 1,2: 57.70685100119465

Qualité de représentation du plan engendré par les axes 1,2,3: 68.98804087497346

8 Qualité de représentation globale :

8.1 Du le plan engendré par les axes 1,2 :

$$I_{123} = \frac{\lambda_1 + \lambda_2}{\sum \lambda_i}$$

$I_{123} = 57,707\%$, donc 57,707% de l'information est contenue dans le plan engendré par les axes 1, 2.

8.2 Du plan engendré par les axes 1,2,3 :

$$I_{123} = \frac{\lambda_1 + \lambda_2 + \lambda_3}{\sum \lambda_i}$$

$I_{123} = 68.988\%$, donc 68.988% de l'information est contenue dans le plan engendré par les axes 1, 2 et 3

9 Qualité de représentation des variables :

	Cos1	Cos2	Cos3
MCSI	0.290217	0.340100	0.046825
BDD	0.651699	0.008557	0.002480
SEC	0.541068	0.004489	0.067282
CPROJ	0.296784	0.165509	0.094467
PROJ	0.028540	0.527325	0.221064
LANG2	0.261801	0.019257	0.324943
ARCH	0.569730	0.066718	0.152371
SYS2	0.676771	0.053298	0.089299
RES2	0.682699	0.009056	0.016577

9.1 Sur le axe 1:

Variable (notes)	Cos1	Qualité de représentation
BDD	0,651	Bien représentée
SEC	0,541	Moyennement représentée
ARCH	0,569	Moyennement représentée
SYS2	0,676	Très bien représentée
RES2	0,682	Très bien représentée

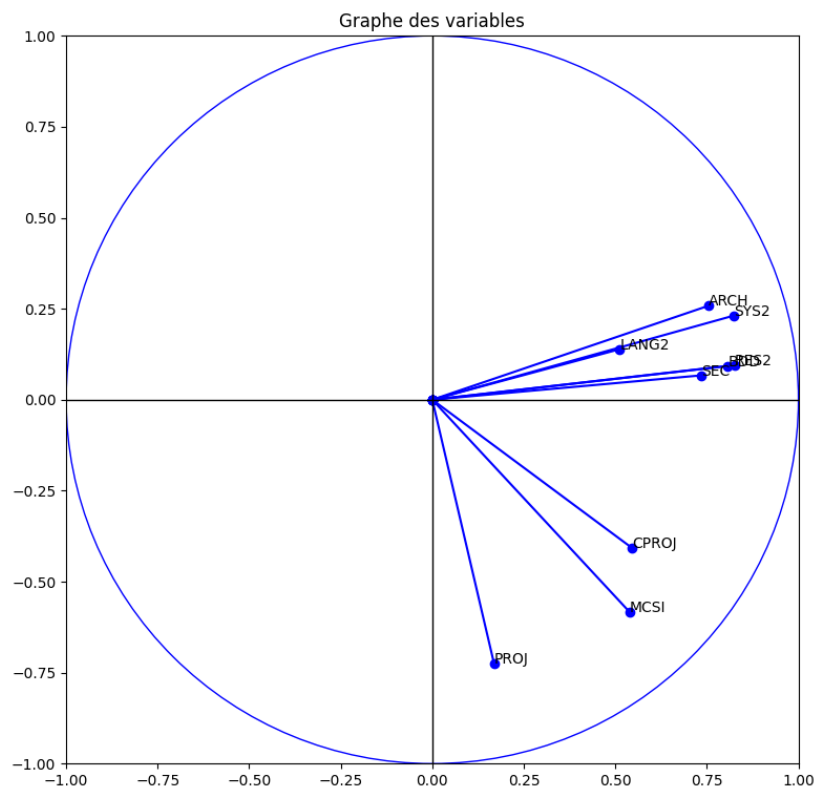
9.2 Sur le axe 2:

Variable (notes)	Cos2	Qualité de représentation
MCSI	0,340	Mal représentée
CPROJ	0,165	Très malement représentée
PROJ	0,527	Moyennement représentée

9.3 Sur le axe 3:

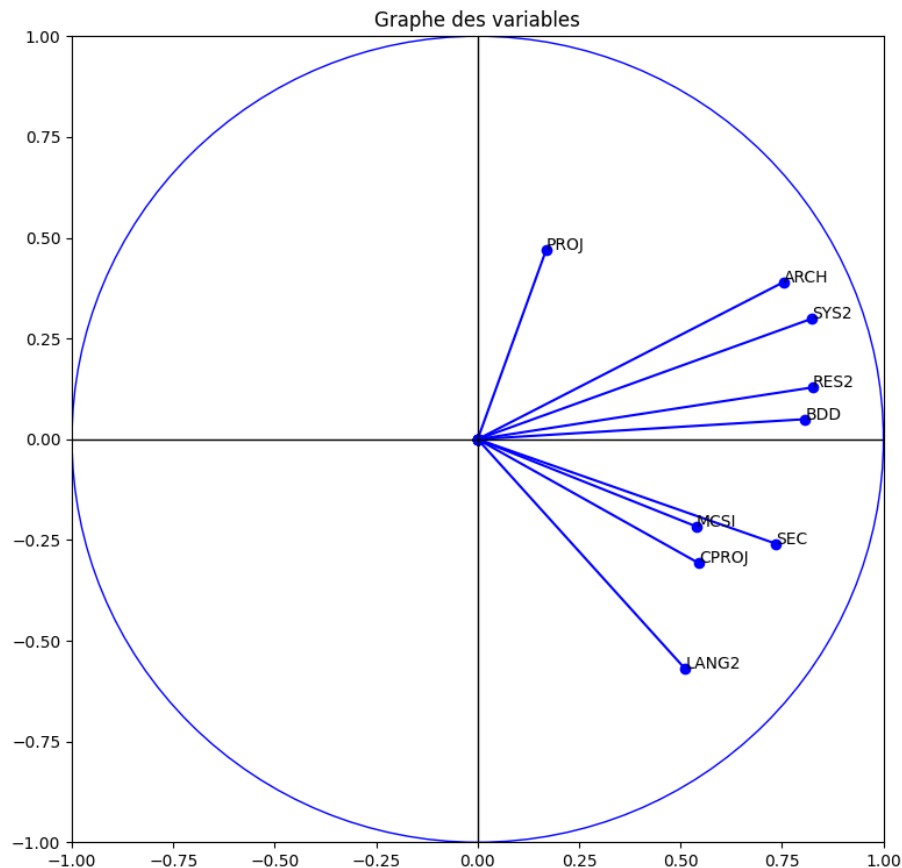
Variable (notes)	Cos2	Qualité de représentation
PROJ	0,221	Très malement représentée
LANG2	0,324	Mal représentée
ARCH	0,152	Très malement représentée

9.4 Sur le plan engendré par les axes 1 et 2:



- La projection des variables SYS2, RES2, SEC, ARCH et BDD sont proches du cercle, alors ces variables sont bien représentées.
- La projection des variables MCSI, CPROJ, LANG2 et PROJ sont positionnés à l'intérieur du cercle, alors ces variables sont moyennement représentées.

9.5 Sur le plan engendré par les axes 1 et 3 :



- La projection des variables SYS2, RES2, SEC, ARCH, BDD et LANG2 sont proches du cercle, alors ces variables sont bien représentées.
- La projection des variables MCSI, CPROJ et PROJ sont positionnés à l'intérieur du cercle, alors ces variables sont moyennement représentées.

10 Contribution relative des individus :

L'individu Fentazi Mohamed Readh est représenté par le rang 32. Nous allons nous contenter de l'étude de notre individus sur ces 2 derniers. Voici à priori les notes de ce dernier:

```
Note_Fentazi = Data_finale.iloc[31,:]
pd.DataFrame(np.array([["MCSI", "BDD", "SEC", "CPROJ", "PROJ", "LANG2", "ARCH", "SYS2", "RES2"],Note_Fentazi]))
```

✓ 0.3s

	0	1	2	3	4	5	6	7	8
0	MCSI	BDD	SEC	CPROJ	PROJ	LANG2	ARCH	SYS2	RES2
1	12.1	13.95	12.42	14.5	13.76	16.02	13.68	11.35	13.65

Sa contribution absolu:

```
contribution_abs_individu = np.sum(Donnes_normalise**2,axis=1)
contribution_abs_fentazi = contribution_abs_individu[31]
contribution_abs_fentazi
```

✓ 0.2s

6.0206436427437415

Une variable X_i contribue à la construction d'un axe α , si sa contribution absolu par rapport à cette axe est supérieure à $1/n$ avec $n = 161$, donc **0.621%**

Sa contribution relative à chaque axe:

```
contribution_rel_individu = np.array(Donnes_ACP**2)
for j in range(9):
    contribution_rel_individu[:,j] = contribution_rel_individu[:,j]/(161*ACP.explained_variance_[j])
contribution_rel_individu = pd.DataFrame(contribution_rel_individu,columns=Label)
contribution_rel_fentazi = pd.DataFrame(np.array([Label[0:3],contribution_rel_individu.iloc[31,0:3]]))
contribution_rel_fentazi
```

✓ 0.3s

	0	1	2
0	CP1	CP2	CP3
1	0.005129777332178406	0.001010722597638001	0.012205778473798782

On peut constater qu'il ne contribue pas assez bien à la construction des axes

Malheureusement l'interprétation va être assez difficile.

11 Qualité de représentation des individus :

Sur le plan engendré par l'axe 1 :

Individus	COS1	Qualité de représentation
I_{32}	0,0051297	Très mal représentée

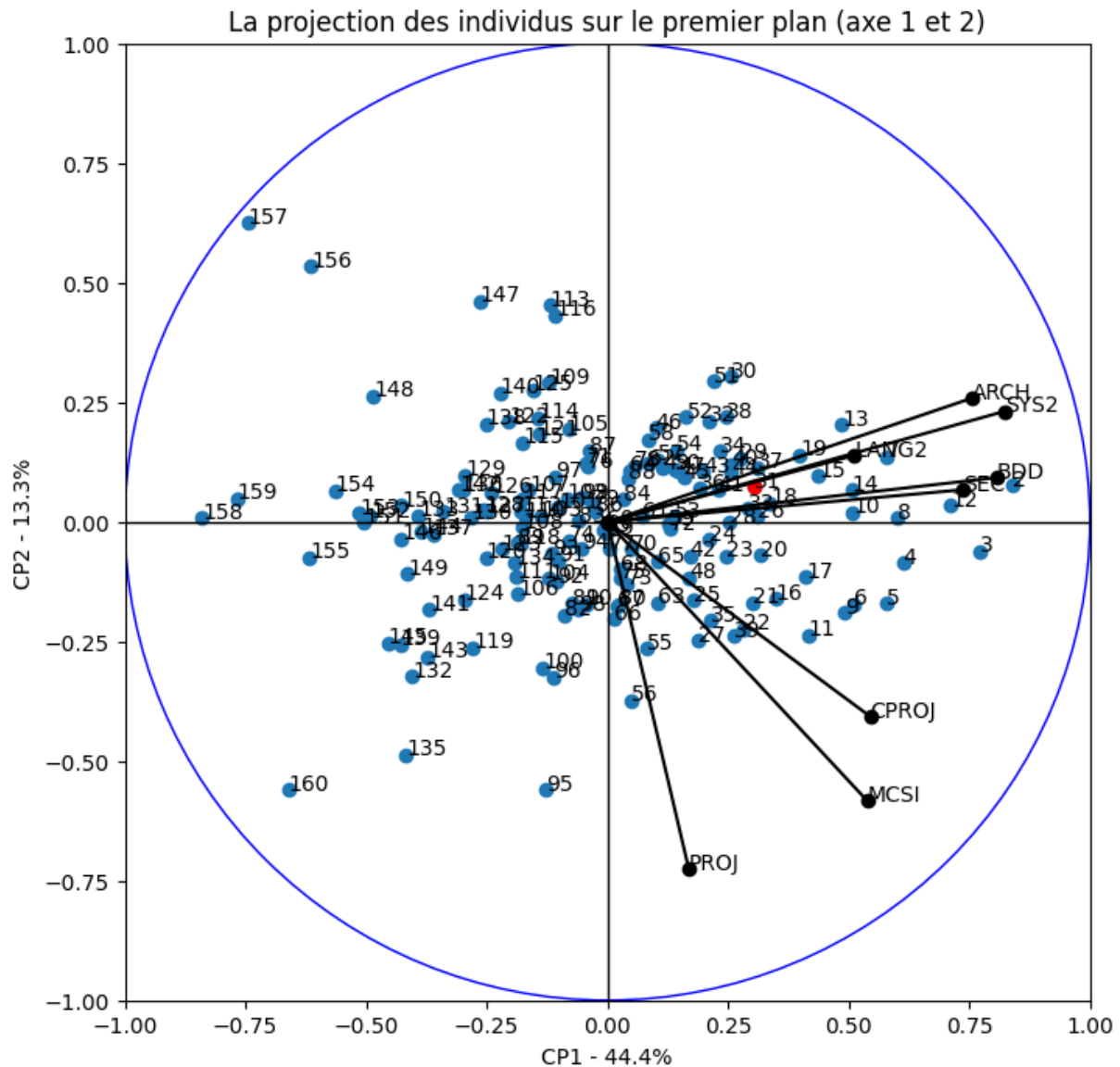
Sur l'axe 2 :

Individus	COS2	Qualité de représentation
I_{32}	0,0010107	Très mal représentée

Sur le plan engendré par les axes 1 et 2 :

Individus	COS12	Qualité de représentation
I_{32}	0,01225	Très mal représentée

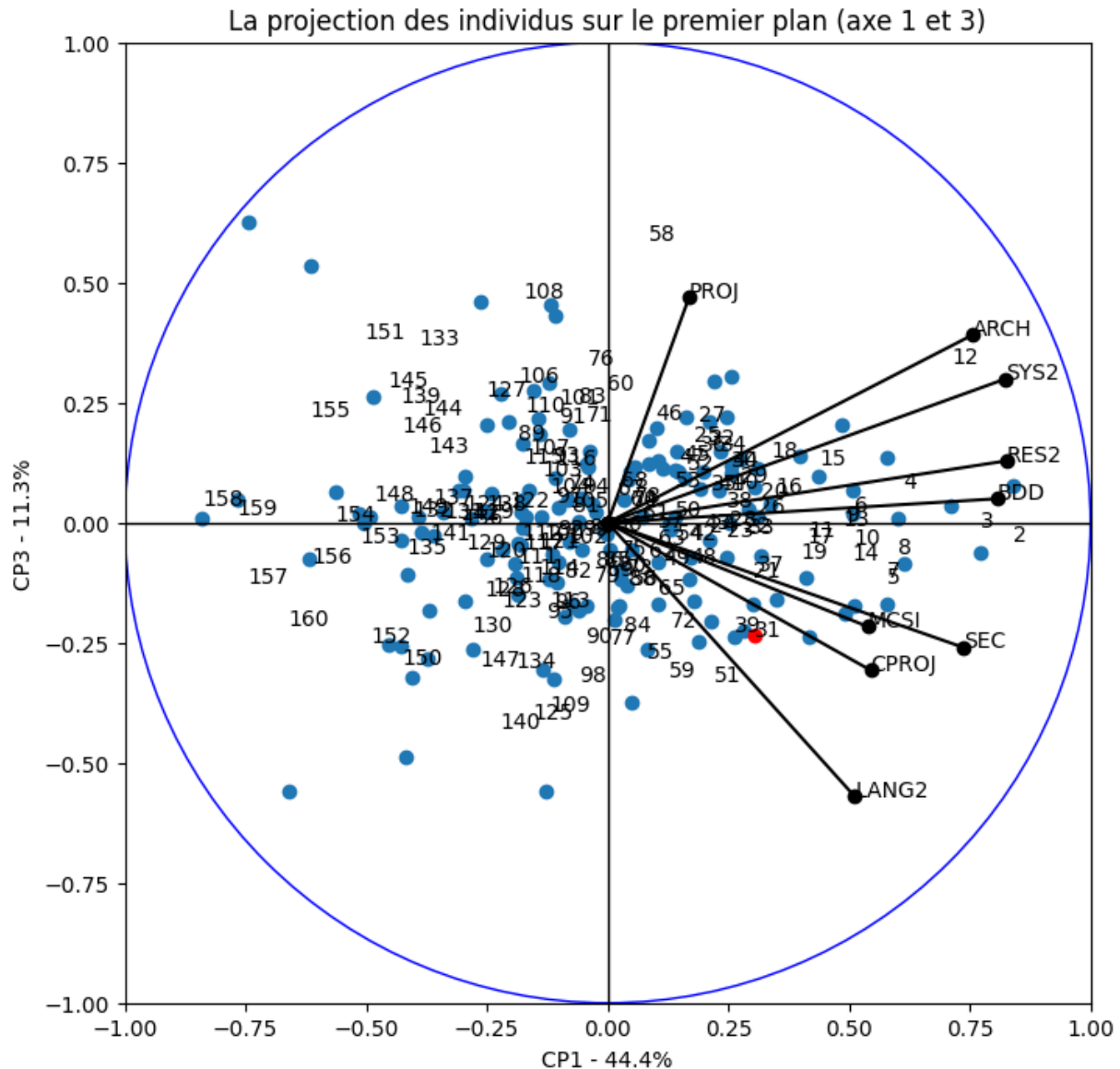
12 Interprétation de l'individus sur le 1er plan (engendré par les axes 1 et 2):



Pour l'individu I_{32} :

- En projetant l'individu sur le 1^{er} axe, on remarque qu'il est très près de l'origine (la moyenne) et se situe dans le côté positif de l'axe. Et on ne peut pas bien l'interpréter, mais on peut en déduire que ce dernier a des notes plus au moins un peu meilleures que la moyenne en **BDD,SYS2,SEC,ARCH** et **RES2** ou il se peut que certain de ces modules en a de bonne note comparé au autre ou il a de mauvaise note.
- En projetant l'individu sur le 2^{ème} axe, on remarque qu'il est très près de l'origine (la moyenne) et se situe dans le côté positif de l'axe. Encore une fois ce n'est possible de l'interpréter, mais on peut conclure que l'individu I_{32} n'a pas obtenu de bonnes notes en CPROJ, PROJ et MCSI comparable à la moyenne

de la promotion sinon y'en a un ou deux d'entre elles qui est mauvaise et l'autre/les autres qui sont bonnes mais le poids de celle faible est un peu plus fort néanmoins on ne peut pas prendre ces résultats car on n'est pas sûr d'elle (information perdu ou individu juste au dessous de la moyenne).



- En projetant sur l'axe 3, notre individu reste près de la moyenne et encore une fois il est dur de l'interpréter.

Tout cela confirme ce qu'on a dit en premier lieu (contribution relative de l'individu) et à cause de la perte d'information (des 6 autres axes) on ne peut pas bien interpréter.

Une question: Que se passerait-il si on interprétait l'individu 32 à lui tout seul?

13 Conclusion :

Grâce à l'acp, on a pu réduire la dimensionnalité du problème de la synthèse de l'information à partir des 9 variables(moyennes des 9 modules) en 3 dimensions avec une précision de 68.988% de l'information totale. Mais on a pu remarquer que:

- Les variables BDD,, SYS2, RES2 et ARCH sont à une certaine degré linéairement dépendantes.
- Les variables CPRJ, MCSI sont eux aussi à une certaine degré linéairement dépendantes

14 Lien utiles:

<https://www.youtube.com/watch?v=Lsue2gEM9D0>