TP – Analyse Factorielle des Correspondances (AFC)

Objectifs pédagogiques

À la fin de ce TP, vous saurez :

1. Construire un **tableau croisé** (tableau de contingence).
2. Lancer une **AFC** en Python.
3. Lire les **sorties principales** : inertie, scree plot, carte Dim1–Dim2.
4. Interpréter les résultats (proximité, association, opposition).

Partie 1 – Préparation des données

1. Importez les bibliothèques nécessaires :

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import prince
```

6. Chargez les données Titanic (déjà incluses dans Seaborn) :

```
titanic = sns.load_dataset('titanic')
titanic.head()
```

   Vérifiez que les colonnes `class` (1st, 2nd, 3rd) et `survived` (0 = No, 1 = Yes) sont bien présentes.

9. Supprimez les lignes manquantes sur ces colonnes :

```
titanic = titanic.dropna(subset=['class','survived'])
```

11. Construisez un **tableau croisé** (*classe* × *survie*) :

```
ct = pd.crosstab(titanic['class'], titanic['survived'])
ct.columns = ['No','Yes']
print(ct)
```

   Question 1. Décrivez ce tableau en mots :

   - Quelle classe a le plus de passagers ?
   - Quelle classe a le plus de survivants ?

Partie 2 – Lancer l'AFC

1. Appliquez l'AFC :

```
ca = prince.CA(n_components=2, random_state=42).fit(ct)
```

3. Obtenez l'inertie expliquée par axe :

```
import numpy as np
```

```
eigen = ca.eigenvalues_              # valeurs propres

inertie = eigen / np.sum(eigen)      # part d'inertie (variance
expliquée)

print("Valeurs propres:", eigen)

print("Inertie par axe:", inertie)

print("Inertie cumulée (2 axes):", inertie[:2].sum())

ca.eigenvalues_ / ca.eigenvalues_.sum()
```

Question 2. Que signifie “inertie” en AFC ?
Question 3. Combien d’axes semble-t-il utile de garder ? Pourquoi ?

4. Obtenez les coordonnées des lignes et colonnes :

```
rows = ca.row_coordinates(ct)    # classes
cols = ca.column_coordinates(ct) # survie
print(rows)
print(cols)
```

Partie 3 – Visualisation

1. **Carte lignes/colonnes (Dim1–Dim2)** :

```
fig, ax = plt.subplots()
ax.axhline(0, lw=1); ax.axvline(0, lw=1)
ax.scatter(rows[0], rows[1], label='Classes')
for i, txt in enumerate(rows.index):
    ax.text(rows.iloc[i,0]*1.05, rows.iloc[i,1]*1.05, txt)
ax.scatter(cols[0], cols[1], marker='s', label='Survie')
for j, txt in enumerate(cols.index):
    ax.text(cols.iloc[j,0]*1.05, cols.iloc[j,1]*1.05, txt)
 ax.set_xlabel("Dim 1"); ax.set_ylabel("Dim 2"); ax.legend()
 ax.set_title("AFC – Titanic (Classe × Survie)")
 plt.show()
```

Question 4. Quelles classes semblent proches de “Yes” ? Quelles classes de “No” ?
Question 5. Quelles classes sont “au centre” (profil moyen, peu discriminant) ?

Le graphique est vide c’est normal vous dessinez des points d’une dimension sur un plan de deux dimensions pour afficher les points

```
import matplotlib.pyplot as plt
# On force Dim2 = 0
x_rows = rows[0]
y_rows = [0]*len(rows)

x_cols = cols[0]
y_cols = [0]*len(cols)

fig, ax = plt.subplots()

# Tracer les classes (cercles bleus)
ax.scatter(x_rows, y_rows, c="blue", label="Classes")
```

```
for i, txt in enumerate(rows.index):
    ax.text(x_rows.iloc[i], 0.05, txt, ha="center", color="blue")
# Tracer la survie (carrés rouges)
ax.scatter(x_cols, y_cols, c="red", marker="s", label="Survie")
for j, txt in enumerate(cols.index):
    ax.text(x_cols.iloc[j], -0.05, txt, ha="center", color="red")
# Habillage
ax.axhline(0, lw=1, color="grey")
ax.set_yticks([])  # enlever l'axe vertical
ax.set_xlabel("Dim 1 (100% inertie)")
ax.set_title("AFC – Titanic (Classe × Survie)")
ax.legend()
plt.show()
```

13. **Scree plot (inertie par axe)** :

```
14. vals = ca.eigenvalues_
15. frac = vals/vals.sum()
16. plt.plot(range(1, len(frac)+1), frac*100, marker='o')
17. plt.xlabel("Axe"); plt.ylabel("Inertie (%)")
18. plt.title("Scree plot – AFC Titanic")
19. plt.show()
```

Question 6. Où se situe le “coude” ? Combien d’axes garder ?

Partie 4 – Qualité de représentation & contributions

1. Calcul du cos² (qualité de représentation) :

```
2. def cos2(coords_df):
3.     sq = coords_df.values**2
4.     tot = sq.sum(axis=1, keepdims=True)
5.     return pd.DataFrame(sq/tot, index=coords_df.index,
   columns=['Dim1','Dim2'])
6. cos2_rows = cos2(rows)
7. cos2_cols = cos2(cols)
8. print("cos² lignes:\n", cos2_rows.round(3))
9. print("cos² colonnes:\n", cos2_cols.round(3))
```

Question 7. Quelles modalités sont bien représentées sur le plan Dim1–Dim2 ?

10. Contribution des lignes à Dim1 (approximation) :

```
11. import numpy as np
12. n = ct.values.sum()
13. r = ct.sum(axis=1)/n  # masses lignes
14. lambda1 = vals[0]
15. contrib_rows_dim1 = (r.values * (rows[0]**2) / lambda1)
16. contrib_rows_dim1 = (contrib_rows_dim1/contrib_rows_dim1.sum())*100
17. print("Contribution lignes Dim1 (%):\n",
    contrib_rows_dim1.round(2))
```

Question 8. Quelles modalités “structurent” Dim1 ?

Partie 5 – Interprétation finale

En 5–10 lignes, rédigez vos conclusions :

- Que raconte **l'axe 1** (opposition principale) ?
- Que raconte **l'axe 2** (nuances) ?
- Quelles classes sont le plus liées à la survie ?
- Quelles sont les modalités les plus discriminantes (fortes contributions) ?
- Quelles modalités sont peu discriminantes (près de l'origine) ?