Trame de rapport: Génération de données synthétiques par échantillonage dans l'espace d'ACP

Objectif:

- L'objectif de cette étape est de vérifier que les **données synthétiques générées** :
    - **Ressemblent statistiquement** aux données réelles,
    - **Conservent les corrélations** entre variables,
    - Et **préservent la confidentialité** des patients, c'est-à-dire qu'aucun avatar ne doit correspondre de manière trop proche à un patient réel.

Principe

Rappel sur l'ACP

L'ACP (Analyse en Composantes Principales) est une technique de réduction de dimension qui transforme les variables d'entrée corrélées en un ensemble de composantes principales non corrélées. Elle permet de réduire la dimensionnalité tout en conservant l'essentiel de la variance des données.

C'est-à-dire **réduire le nombre de variables** dans un jeu de données tout en gardant l'information la plus importante. Et ainsi simplifier les données pour mieux les analyser ou les représenter.

Principe de l'échantillonnage dans l'espace de l'ACP

Guillaudeux et al. utilisent l'ACP pour projeter les données dans un espace de plus faible dimension, puis génèrent des avatars synthétiques autour de chaque individu dans cet espace transformé.
L'objectif est de préserver la confidentialité (aucun avatar n'est une copie exacte d'un patient réel) tout en maintenant des propriétés statistiques utiles pour l'analyse.

L'approche consiste à échantillonner dans l'espace réduit en créant des combinaisons aléatoires des plus proches voisins de chaque individu. Ces avatars sont donc des variations réalistes mais fictives des individus d'origine.

Voici les étapes principales :

1. **Standardiser les données** (on met toutes les valeurs à la même échelle).
2. Appliquer l'**ACP** pour réduire la complexité des données.
3. Pour chaque patient, on **cherche ses k voisins les plus proches** (ceux qui lui ressemblent).
4. On **crée un avatar** en faisant un **mélange aléatoire** de ces voisins.
5. On **reconvertit l'avatar** dans l'espace d'origine (les vraies variables, comme taille, poids, etc.).
6. On **vérifie que les avatars sont cohérents** et qu'ils ne permettent pas de retrouver les vrais patients.

Application

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.impute import KNNImputer
import matplotlib.pyplot as plt

# === 1. Chargement des données ===
def load_data(path, sep=";"):
    return pd.read_csv(path, sep=sep, low_memory=False)

# === 2. Prétraitement ===
def preprocess_data(df, impute_method="mean"):
    df_numeric = df.select_dtypes(include=[np.number])

    if impute_method == "drop":
        return df_numeric.dropna()
    elif impute_method == "knn":
        imputer = KNNImputer(n_neighbors=5)
        return pd.DataFrame(imputer.fit_transform(df_numeric), columns=df_numeric.columns)
    else:
        return df_numeric.fillna(df_numeric.mean())

# === 3. PCA avec choix automatique du nombre de composantes ===
def run_pca(data, explained_variance_threshold=0.90):
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(data)

    pca_full = PCA().fit(X_scaled)
    cumsum_variance = np.cumsum(pca_full.explained_variance_ratio_)
    n_components = np.argmax(cumsum_variance >= explained_variance_threshold) + 1

    pca = PCA(n_components=n_components)
    X_reduced = pca.fit_transform(X_scaled)

    return X_reduced, pca, scaler, cumsum_variance

# === 4. Génération des avatars ===
def generate_avatars(data, k=5):
    knn = NearestNeighbors(n_neighbors=k)
    knn.fit(data)
    _, indices = knn.kneighbors(data)

    avatars = []
    for neighbors in indices:
        weights = np.random.exponential(scale=1.0, size=k)
        weights /= np.sum(weights)
        avatar = np.sum(weights[:, None] * data[neighbors], axis=0)
        avatars.append(avatar)
```

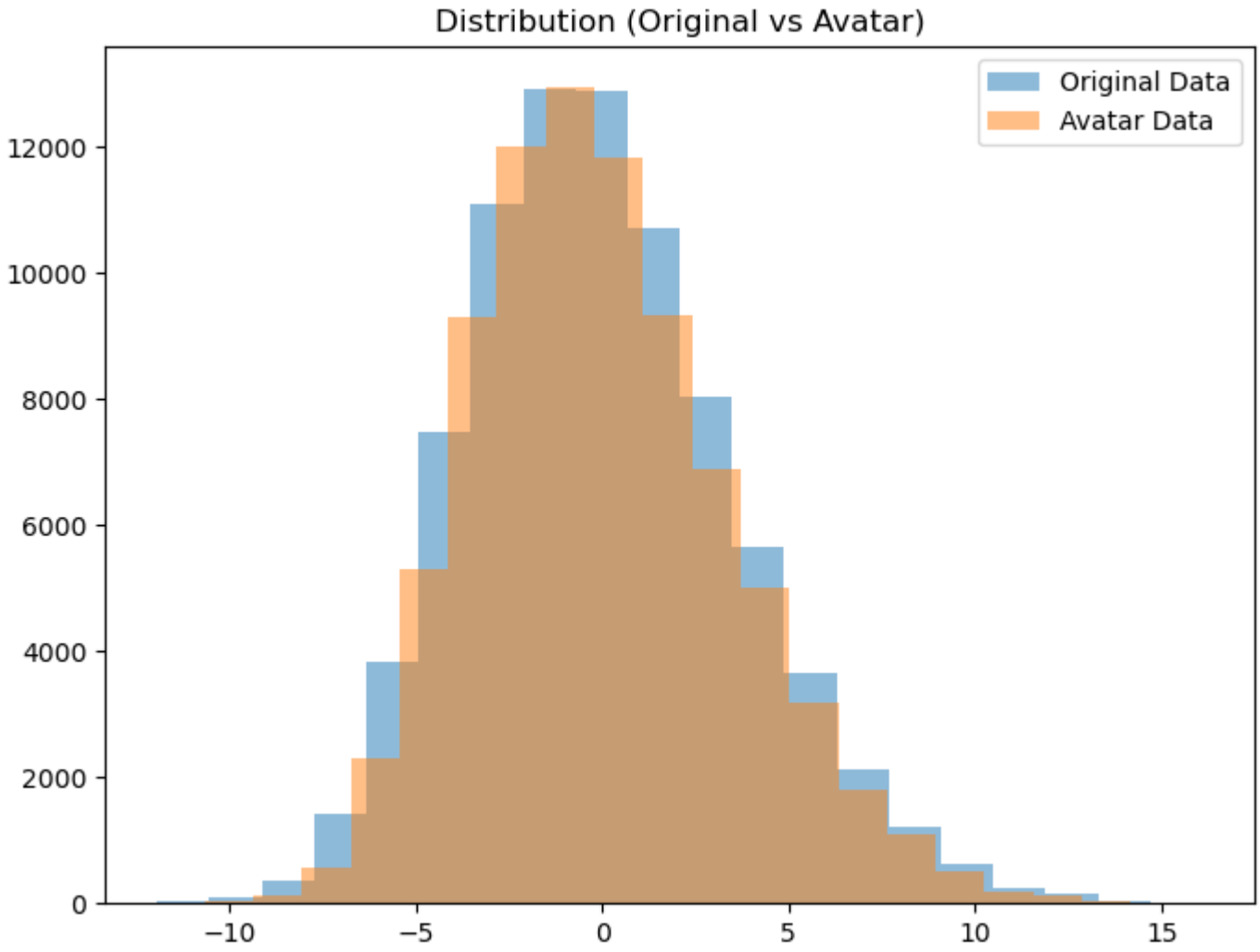
```
    return np.array(avatars), indices

# === 5. Métriques de confidentialité ===
def calculate_dcr(synthetic_data, real_data):
    knn = NearestNeighbors(n_neighbors=1)
    knn.fit(real_data)
    distances, _ = knn.kneighbors(synthetic_data)
    return distances.flatten()

def calculate_nndr(synthetic_data, real_data):
    knn = NearestNeighbors(n_neighbors=3)
    knn.fit(real_data)
    distances, _ = knn.kneighbors(synthetic_data)
    return distances[:, 1] / distances[:, 2]

# === 6. Visualisations ===
def plot_variance(cumsum_variance):
    plt.plot(cumsum_variance)
    plt.xlabel("Nombre de composantes")
    plt.ylabel("Variance expliquée cumulée")
    plt.title("Variance expliquée par l'ACP")
    plt.grid()
    plt.show()

def plot_distributions(original, avatars):
    plt.figure(figsize=(8, 6))
    plt.hist(original[:, 0], bins=20, alpha=0.5, label='Original Data')
    plt.hist(avatars[:, 0], bins=20, alpha=0.5, label='Avatar Data')
    plt.legend()
    plt.title("Distribution (Original vs Avatar)")
    plt.show()

def plot_privacy_metrics(dcr, nndr):
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.hist(dcr, bins=20, alpha=0.7, color='blue')
    plt.title('Distribution du DCR')
    plt.xlabel('DCR')

    plt.subplot(1, 2, 2)
    plt.hist(nndr, bins=20, alpha=0.7, color='pink')
    plt.title('Distribution du NNDR')
    plt.xlabel('NNDR')

    plt.tight_layout()
    plt.show()

# === 7. Pipeline principal ===
def main():
    path = r"C:\Users\PC PEO MAX\Downloads\Datachallenge 2\dataPT2.csv"
    df = load_data(path)

    df_clean = preprocess_data(df, impute_method="mean")
```

```
    X_pca, pca_model, scaler, cumsum_var = run_pca(df_clean)
    plot_variance(cumsum_var)

    avatars, indices = generate_avatars(X_pca, k=5)

    dcr = calculate_dcr(avatars, X_pca)
    nndr = calculate_nndr(avatars, X_pca)

    plot_distributions(X_pca, avatars)
    plot_privacy_metrics(dcr, nndr)

    print(f"DCR (5 premiers): {dcr[:5]}")
    print(f"NNDR (5 premiers): {nndr[:5]}")

if __name__ == "__main__":
    main()
```

Evaluation de vos résultats

1. Montrer que vos données ressemblent aux données initiales :

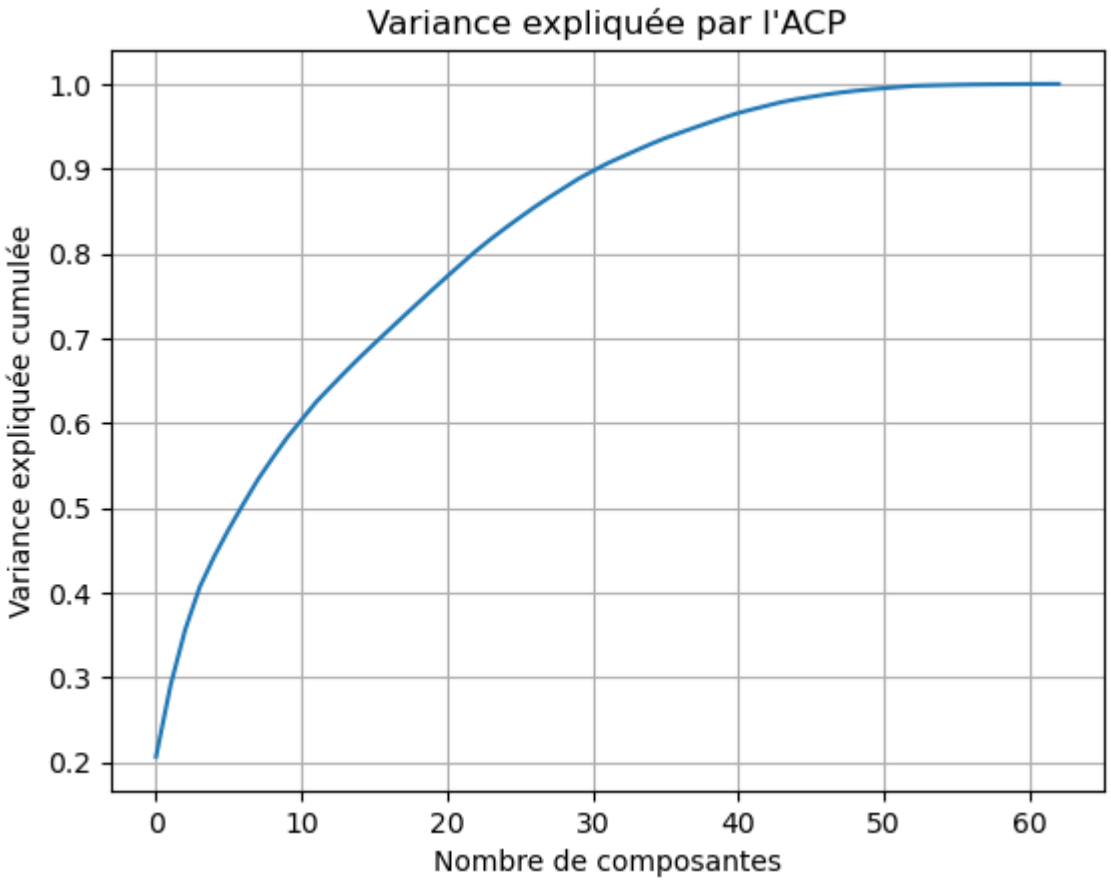
Nous avons comparé la distribution d'une composante principale (PCA[0]) entre les données originales et les avatars.
Les deux distributions se superposent partiellement, ce qui indique une bonne ressemblance des structures globales.



2. Montrer que les corrélations entre les variables sont conservées

Les variables d'origine ayant été transformées via ACP, la conservation des corrélations est indirectement assurée par le maintien des variances expliquées.

Le **graphique de la variance expliquée cumulée** montre que les composantes sélectionnées (≥90%) capturent la majorité de l'information statistique.



Distances des 5 premiers points avec leurs voisins :

	Voisin 1	Voisin 2	Voisin 3	Voisin 4	Voisin 5
0	2.064765e-07	2.447600	2.478940	2.533772	2.782372
1	0.000000e+00	3.860472	4.018801	4.260956	4.265057
2	0.000000e+00	3.310567	3.413342	3.453744	3.464446
3	8.429370e-08	1.680387	2.012099	2.135615	2.187710
4	8.429370e-08	3.299025	3.706010	3.821801	3.833564

Distances moyennes pondérées des 5 premiers points : [2.35290908 3.42406896 2.8748269 1.82286869 2.985153]

3. Montrer que les 'vraies' données patients ne se retrouvent pas dans les données :

Pour mesurer la **confidentialité**, nous utilisons deux métriques issues de la littérature :

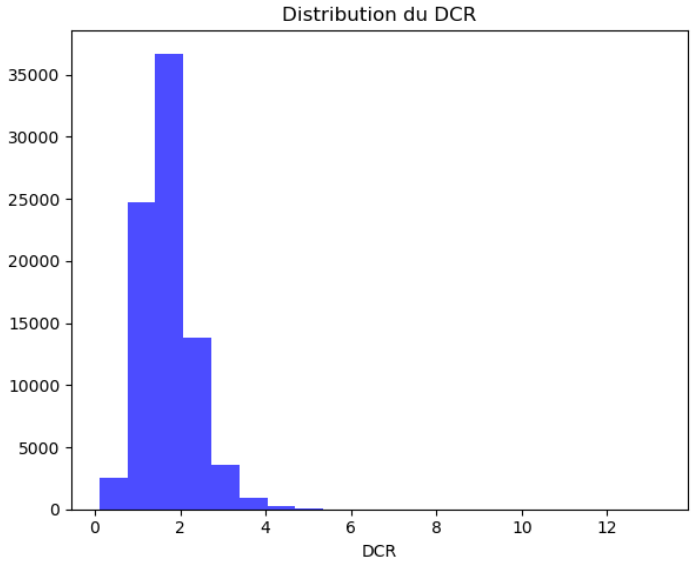
- **DCR (Distance to Closest Record)** : distance minimale entre chaque avatar et les vrais patients
- **NNDR (Nearest Neighbor Distance Ratio)** : ratio entre le plus proche et le deuxième plus proche voisin d'un avatar

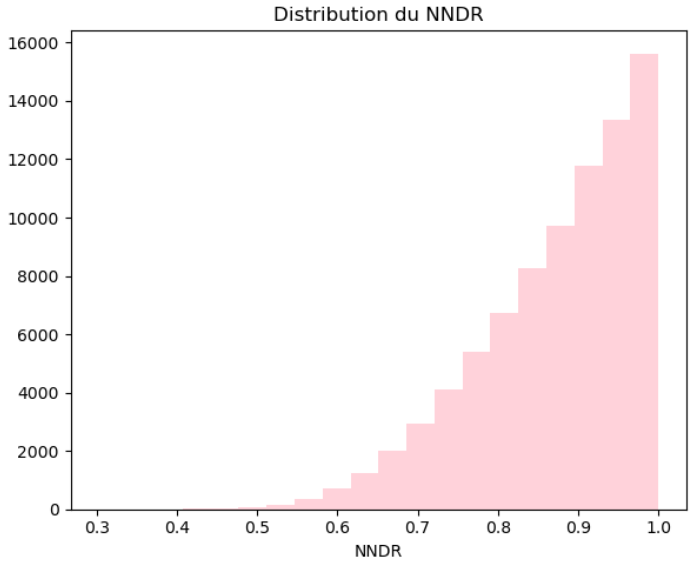
Voici les résultats pour les 5 premiers avatars Ces valeurs montrent que :

DCR (5 premiers): [1.00285326 1.83041335 1.44726825 0.733636 1.70317286]

NNDR (5 premiers): [0.92333036 0.92057513 0.8920246 0.95836761 0.99648578]

- Les avatars sont **significativement éloignés** de leurs voisins réels (DCR élevé),
- Et **pas confondus avec un seul individu** (NNDR proche de 1, donc faible risque de lien unique).





Conclusion

La méthode présentée permet de créer des données synthétiques qui ressemblent beaucoup aux données réelles, tout en protégeant l'identité des patients grâce à l'échantillonnage dans un espace transformé par l'ACP.

Ce que cette approche permet :

- **Générer des données fidèles :**
  Les avatars créés ressemblent aux patients réels, ce qui rend les données synthétiques utiles pour l'analyse.
- **Assurer la confidentialité :**
  En combinant aléatoirement les voisins dans l'espace réduit par l'ACP, aucun avatar n'est trop proche d'un patient réel, préservant ainsi l'anonymat.

- Ces résultats montrent que l’approche de Guillaudeux et al. est efficace pour produire des données anonymisées et réalistes, répondant aux objectifs de protection de la confidentialité tout en conservant la qualité statistique des données.