ANNEXE

DADDIO Melanie - FIX Emma - MARZOUK Moustafa - RAKOTOMANANA Zava

Identifier quels pays s'en sortent relativement bien et lesquels nécessitent une attention immédiate ?

1 Installation des packages nécessaires à l'étude

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import pearsonr

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

from skimpy import skim

# from pingouin import partial_corr
```

2 Import des données

[ ]:
```
#import
CovidCases = pd.read_csv("CovidCases.csv", sep=",")

CovidCases
```

3 Compréhension des données

[ ]:
```
skim(CovidCases)
```

[ ]:
```
# vérifier qu'il n'y ai pas de valeurs "bizarre"
CovidCases[CovidCases['TotalCases']<CovidCases["TotalDeaths"]]
```

[ ]:
```
CovidCases[CovidCases['TotalCasesPerMillion']<CovidCases["TotalDeathsPerMillion"]]
```

[ ]:
```
CovidCases[CovidCases['TotalPopulation']<CovidCases["TotalCases"]]
```

[ ]:
```
CovidCases[CovidCases['TotalPopulation']<CovidCases["TotalRecovered"]]
```

Il ne semble pas avoir des valeurs “bizarre”.

3.1 Visualisation des données :

[ ]:
```
continue_var = CovidCases.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()
categorical_var = CovidCases.select_dtypes(include=['object']).columns.tolist()
```

3.1.1 Visualisation des variables continues :

[ ]:
```
# On affiche les histogrammes des variables continues

n_cols = 3  # Nombre de colonnes par ligne
n_rows = (len(continue_var) + n_cols - 1) // n_cols  # Calculer le nombre de␣
 ↪lignes nécessaires

plt.figure(figsize=(n_cols * 5, n_rows * 5))

for i, col in enumerate(continue_var[1:]):
    plt.subplot(n_rows, n_cols, i + 1)
    sns.histplot(CovidCases[col], color='skyblue', bins=20)
    plt.title(f'Histogramme de {col}')
    plt.xlabel(col)
    plt.ylabel('Fréquence')

plt.tight_layout()
plt.show()
```

[ ]:
```
# BOXPLOT
n_cols = 3
n_rows = (len(continue_var) + n_cols - 1) // n_cols

plt.figure(figsize=(n_cols * 5, n_rows * 5))
```

```
for i, col in enumerate(continue_var[1:]):
    plt.subplot(n_rows, n_cols, i + 1)
    sns.boxplot(y=CovidCases[col], palette='Set2')
    plt.title(f'Boxplot de {col}')
    plt.ylabel(col)

plt.tight_layout()

plt.show()
```

[ ]:
```
# Visualisationdes valaurs extremes heatmap
CC = CovidCases.iloc[:,2:]
CC_c = CC.sub(CC.mean())
CC_cr = CC_c.div(CC_c.std())
CC_cr = CC_cr.T
sns.set()
sns.heatmap(CC_cr,cmap="viridis")
plt.show()
```

3.1.2 Visualistion des variables continues en fonction des pays :

[ ]:
```
#barplot
#nombre total de cas en fonction du pays
plt.figure(figsize=(14, 7))
sns.barplot(x='TotalDeaths', y='Country', data=CovidCases.
 ↪sort_values('TotalDeaths', ascending=False))
plt.title('Total Death of COVID-19 in Asian Countries')
plt.xlabel('Total Death')
plt.ylabel('Country')
plt.show()

#nombre de cas par million d'individus en fonction du pays
plt.figure(figsize=(14, 7))
sns.barplot(x='TotalDeathsPerMillion', y='Country', data=CovidCases.
 ↪sort_values('TotalDeathsPerMillion', ascending=False))
plt.title('Total Death of COVID-19 in Asian Countries per million')
plt.xlabel('Total Death per million')
plt.ylabel('Country')
plt.show()
```

[ ]:
```
#nombre total de mort en fonction du pays
plt.figure(figsize=(14, 7))
sns.barplot(x='TotalDeaths', y='Country', data=CovidCases.
 ↪sort_values('TotalDeaths', ascending=False))
plt.title('Total deaths of COVID-19 in asian countries')
plt.xlabel('Total deaths')
```

```
plt.ylabel('Country')
plt.show()

#nombre de mort par million d'individus en fonction du pays
plt.figure(figsize=(14, 7))
sns.barplot(x='TotalDeathsPerMillion', y='Country', data=CovidCases.
 ↪sort_values('TotalDeathsPerMillion', ascending=False))
plt.title('Total deaths of COVID-19 in asian countries per million')
plt.xlabel('Total deaths per million')
plt.ylabel('Country')
plt.show()
```

[ ]:
```
#nombre total de test en fonction du pays
plt.figure(figsize=(14, 7))
sns.barplot(x='TotalTests', y='Country', data=CovidCases.
 ↪sort_values('TotalTests', ascending=False))
plt.title('Total tests of COVID-19 in asian countries')
plt.xlabel('Total tests')
plt.ylabel('Country')
plt.show()

#nombre de test par million d'individus en fonction du pays
plt.figure(figsize=(14, 7))
sns.barplot(x='TotalTestsPerMillion', y='Country', data=CovidCases.
 ↪sort_values('TotalTestsPerMillion', ascending=False))
plt.title('Total tests of COVID-19 in asian countries per million')
plt.xlabel('Total tests per million')
plt.ylabel('Country')
plt.show()
```

3.2 Analyse de relation entre variables :

[ ]:
```
#plot de x en fonction de y (pour préparer une éventuelle prédiction)
sns.pairplot(CovidCases.iloc[:,2:])
```

On a decidé de ne pas preter plus d'attention aux pairplot car nous ne remarquons pas un tendance particulière.

[ ]:
```
# matrice correlation
corr_matrix = CovidCases.iloc[:,2:].corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Matrice de Corrélation des Variables COVID-19")
plt.show()
```

4 Données manquantes

En regardant sur le site on remarque que jusqu'à mi 2022 le Macao avait un total de mort déclaré à 0. Pour se pays nous imputerons donc les valeurs manquantes "totaldeath" et "totaldeathpermillion" à 0.

Pour le Tajikistan nous n'avons pas trouvé le nombre de tests dans le site nous allons donc emputer par la mediane. De plus la variable test ne nous a pas parue tres pertinente pour la suite de notre analyse et nous allons surment pas l'utiliser.

[ ]:
```
CovidCases.loc[CovidCases["Country"] == "Macao", :] = CovidCases.
 ↪loc[CovidCases["Country"] == "Macao", :].fillna(0)
```

[ ]:
```
print(CovidCases.isnull().sum())

#imputation par médiane
CovidCases.fillna(CovidCases.median(numeric_only=True), inplace=True)

print(CovidCases.isnull().sum())
```

5 Tests d'independance : test de corrélation de pearson

[ ]:
```
#TotalCases vs TotalDeaths
print(pearsonr(CovidCases.TotalCases, CovidCases.TotalDeaths))

#TotalCasesPerMillion vs TotalDeathsPerMillion
print(pearsonr(CovidCases.TotalCasesPerMillion, CovidCases.
 ↪TotalDeathsPerMillion))

#TotalCases vs TotalDeathsPerMillion
print(pearsonr(CovidCases.TotalCases, CovidCases.TotalDeathsPerMillion))

#TotalCases vs TotalCasesPerMillion
print(pearsonr(CovidCases.TotalCases, CovidCases.TotalCasesPerMillion))
```

[ ]:
```
# Corrélation partielle en contrôlant la population
# corr_p = partial_corr(data=CovidCases, x='TotalCases',␣
 ↪y='TotalCasesPerMillion', covar='TotalPopulation', method='pearson')
# print(corr_p)
```

6 Tests de redondance

[ ]:
```
# tester si une variables est une combinaison lineaire c'autres variables
q = CovidCases.select_dtypes(include=[np.number])
q = q.iloc[:,1:]
for col in q.columns:
```

```
    X = q.drop(columns=[col])
    y = q[col]
    model = LinearRegression()
    model.fit(X, y)
    r2 = model.score(X, y)

    if r2 > 0.95:
        print(f" {col} est très bien prédite par les autres variables (R² = {r2:
↪.3f}), elle est redondante")
```

On voit bien que les variables qui existent avec “PerMillion” et celles qui existent sans qui sont redondantes, ce qui est totalemnt logique.

7 Étude de taux

7.1 Calculs des taux et visualisation du classement des pays

[ ]:
```
# taux de mortalité
CovidCases['TauxMortalite'] = CovidCases['TotalDeathsPerMillion'] /␣
↪CovidCases['TotalCasesPerMillion'] * 100
plt.figure(figsize=(10,8))
sns.barplot(x='TauxMortalite', y='Country', data=CovidCases.
↪sort_values('TauxMortalite', ascending=False))
plt.ylabel('Pays')
plt.xlabel('Taux de mortalité')
plt.title("Taux de mortalité des pays d'Asie face au Covid-19")
plt.show()
```

[ ]:
```
# taux de récupération (on calcul le par million nous même)
CovidCases["TotalRecoveredPerMillion"] = (CovidCases["TotalRecovered"]␣
↪*1000000)/CovidCases["TotalPopulation"]
CovidCases["TauxRecuperation"] = CovidCases['TotalRecoveredPerMillion'] /␣
↪CovidCases['TotalCasesPerMillion'] * 100
plt.figure(figsize=(10,8))
sns.barplot(x='TauxRecuperation', y='Country', data=CovidCases.
↪sort_values('TauxRecuperation', ascending=False))
plt.ylabel('Pays')
plt.xlabel('Taux de récupération')
plt.title("Taux de récupération des pays d'Asie face au Covid-19")
plt.show()
```

[ ]:
```
# taux de cas actif
CovidCases["TauxCasActifs"] = CovidCases['ActiveCases'] /␣
↪CovidCases['TotalCases'] * 100
plt.figure(figsize=(10,8))
```

```
sns.barplot(x='TauxCasActifs', y='Country', data=CovidCases.
 ↪sort_values('TauxCasActifs', ascending=False))
plt.ylabel('Pays')
plt.xlabel('Taux de cas actifs')
plt.title("Taux de cas actifs dans les pays d'Asie (dernière données rentrées)")
plt.show()
```

7.2 Calculs des seuils et recupération des pays en difficulté

[ ]:
```
# Pays qui sont en dessous/au dessus d'un seuil
pays_cas = CovidCases[(CovidCases['TotalCasesPerMillion'] >␣
 ↪CovidCases['TotalCasesPerMillion'].quantile(0.9))]
pays_mortalite = CovidCases[(CovidCases['TauxMortalite'] >␣
 ↪CovidCases['TauxMortalite'].quantile(0.9))]
pays_test = CovidCases[(CovidCases['TotalTestsPerMillion'] <␣
 ↪CovidCases['TotalTestsPerMillion'].quantile(0.1) )]

print("Liste des pays au dessus du seuil de cas : \n\n", pays_cas[['Country',␣
 ↪'TotalCasesPerMillion', 'TotalTestsPerMillion', 'TauxMortalite']])
print("\nListe des pays au dessus du seuil de mortalité : \n\n",␣
 ↪pays_mortalite[['Country', 'TotalCasesPerMillion', 'TotalTestsPerMillion',␣
 ↪'TauxMortalite']])
print("\nListe des pays en dessous du seuil de tests : \n\n",␣
 ↪pays_test[['Country', 'TotalCasesPerMillion', 'TotalTestsPerMillion',␣
 ↪'TauxMortalite']])
```

[ ]:
```
# Pays au dessus/en dessous de plusieurs seuils
pays_concat = pd.concat([pays_cas['Country'], pays_mortalite['Country'],␣
 ↪pays_test['Country']])
pays_counts = pays_concat.value_counts()
pays_critique = pays_counts[pays_counts >= 2].index.tolist()

print("\nListe des pays qui dépassent plusieurs seuils : \n\n", CovidCases.
 ↪loc[CovidCases['Country'].isin(pays_critique), ['Country',␣
 ↪'TotalCasesPerMillion', 'TotalTestsPerMillion', 'TauxMortalite']])
```

8 PCA

8.1 TEST 1 PCA avec les variables : “ActiveCasesPerMillion”, “TauxMortalite”

[ ]:
```
# Préparation nous le PCA
CovidCases = pd.read_csv("CovidCases.csv", sep=",")
CovidCases.loc[CovidCases["Country"] == "Macao", :] = CovidCases.
 ↪loc[CovidCases["Country"] == "Macao", :].fillna(0)
CovidCases = CovidCases.drop(columns=["ID"])
```

```
CovidCases['ActiveCasesPerMillion'] = CovidCases['ActiveCases'] * 1000000 /␣
 ↪CovidCases['TotalPopulation']
colums_stand = CovidCases.columns[1:]
print(colums_stand)

# Calcul taux mortalité
CovidCases['TauxMortalite'] = CovidCases['TotalDeathsPerMillion'] /␣
 ↪CovidCases['TotalCasesPerMillion'] * 100
```

- Standardiser les données

[ ]:
```
scaler = StandardScaler()
CovidCases[colums_stand] = scaler.fit_transform(CovidCases[colums_stand])
CovidCases.head()
```

- Selection des variables

[ ]:
```
CovidCases_without_country = CovidCases[["ActiveCasesPerMillion",␣
 ↪"TauxMortalite"]]
CovidCases_without_country.head()
```

[ ]:
```
X = CovidCases_without_country.iloc[:, :].values
```

[ ]:
```
X
```

- PCA :

[ ]:
```
pca = PCA()
X_pca=pca.fit_transform(X)
```

[ ]:
```
# Inertie expliquée
explained_variance = pca.explained_variance_ratio_ * 100
explained_variance
```

Choix du nombre d'axes :

[ ]:
```
# Analyse des valeurs propres
n_components = len(pca.explained_variance_)
comp = pd.DataFrame(
    {
        "Dimension" : ["Dim" + str(x + 1) for x in range(n_components)],
        "Valeur propre" : pca.explained_variance_,
        "% variance expliquée" : np.round(pca.explained_variance_ratio_ * 100),
        "% cum. var. expliquée" : np.round(np.cumsum(pca.
 ↪explained_variance_ratio_) * 100)
    },
    columns = ["Dimension", "Valeur propre", "% variance expliquée", "% cum.␣
 ↪var. expliquée"]
```

```
)
comp
```

[ ]:
```
# Scree plot pour choisir le nombre de composantes principales
g_comp = sns.barplot(x = "Dimension",
                     y = "% variance expliquée",
                     palette = ["lightseagreen"],
                     data = comp)
g_comp.set(ylabel = "Variance expliquée (%)")
g_comp.figure.suptitle("Variance expliquée par dimension")
plt.axhline(y = 25, linewidth = .5, color = "dimgray", linestyle = "--") # 25 =␣
 ↪100 / 4 (nb dimensions)
plt.text(3.25, 26, "25%")
```

[ ]:
```
# Calcul du cosinus carré des variables
cos_squared = np.square(pca.components_)
df_cos_squared = pd.DataFrame(cos_squared, columns=['PC{}'.format(i+1) for i in␣
 ↪range(n_components)])
df_cos_squared.index = CovidCases_without_country.columns

print(df_cos_squared)
```

[ ]:
```
# Contribution à la formation de l'axe
components = pca.components_

n_components = X.shape[1]
feature_names=CovidCases_without_country.columns

loadings = pca.components_.T
eigenvalues = pca.explained_variance_
variable_contributions = (loadings**2) * eigenvalues

column_names = [f'PC{i+1}_contrib' for i in range(n_components)]
variable_contrib_df = pd.DataFrame(variable_contributions,␣
 ↪columns=column_names, index=feature_names)

variable_contrib_df
```

[ ]:
```
# Calculer la contribution des individus à la formation des axes
eigenvalues = pca.explained_variance_
contributions = (X_pca**2) / (X_pca.shape[0] * eigenvalues)
contrib_percent = contributions * 100

column_names = [f'PC{i+1}_contrib' for i in range(n_components)]
contrib_df = pd.DataFrame(contrib_percent, columns=column_names)

print(contrib_df)
```

[ ]:
```
# Créer le cercle de corrélation
coeff = np.transpose(pca.components_[0:2, :])
n = coeff.shape[0]
xs = np.array([1, 0])
ys = np.array([0, 1])

plt.figure(figsize=(10, 10))
for i in range(n):
    plt.arrow(0, 0, coeff[i, 0], coeff[i, 1], color='k', alpha=0.9,␣
 ↪head_width=0.02)
    plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, feature_names[i],␣
 ↪color='k', ha='center', va='center')

circle = plt.Circle((0, 0), 1, color='gray', fill=False, linestyle='--')
plt.gca().add_artist(circle)
plt.xlim(-1, 1)
plt.ylim(-1, 1)
plt.axhline(0, color='gray', linewidth=1)
plt.axvline(0, color='gray', linewidth=1)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Cercle de corrélation')
plt.show()
```

[ ]:
```
# Projection des pays
plt.figure(figsize=(12, 7))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)

for i, country in enumerate(CovidCases.iloc[:, 0]):
    plt.text(X_pca[i, 0], X_pca[i, 1], country, fontsize=9, alpha=0.7)

plt.xlabel("PC1 ({}% variance)".format(round(explained_variance[0], 1)))
plt.ylabel("PC2 ({}% variance)".format(round(explained_variance[1], 1)))
plt.title("Projection des pays sur le premier plan factoriel")
plt.axhline(0, color="grey", linestyle="--", linewidth=0.8)
plt.axvline(0, color="grey", linestyle="--", linewidth=0.8)
plt.grid()
plt.show()
```

[ ]:
```

```

8.2 TEST 1 bis PCA avec les variables : "ActiveCasesPerMillion", "TauxMortalite" (sans le Yemen et Cyprus)

[ ]:
```
# Préparation nous le PCA
CovidCases = pd.read_csv("CovidCases.csv", sep=",")
CovidCases.loc[CovidCases["Country"] == "Macao", :] = CovidCases.
 ↪loc[CovidCases["Country"] == "Macao", :].fillna(0)
CovidCases = CovidCases.drop(columns=["ID"])
CovidCases['ActiveCasesPerMillion'] = CovidCases['ActiveCases'] * 1000000 /␣
 ↪CovidCases['TotalPopulation']
colums_stand = CovidCases.columns[1:]
print(colums_stand)

# Calcul taux mortalité
CovidCases['TauxMortalite'] = CovidCases['TotalDeathsPerMillion'] /␣
 ↪CovidCases['TotalCasesPerMillion'] * 100

# Supprimer Yemen et cyprus
CovidCases = CovidCases[CovidCases['Country'] != 'Yemen']
CovidCases = CovidCases[CovidCases['Country'] != 'Cyprus']
print(CovidCases.shape)
```

- Standardiser les données

[ ]:
```
scaler = StandardScaler()
CovidCases[colums_stand] = scaler.fit_transform(CovidCases[colums_stand])
CovidCases.head()
```

- Selection des variables

[ ]:
```
CovidCases_without_country = CovidCases[["ActiveCasesPerMillion",␣
 ↪"TauxMortalite"]]
CovidCases_without_country.head()
```

[ ]:
```
X = CovidCases_without_country.iloc[:, :].values
```

[ ]:
```
X
```

- PCA :

[ ]:
```
pca = PCA()
X_pca=pca.fit_transform(X)
```

[ ]:
```
# Inertie expliquée
explained_variance = pca.explained_variance_ratio_ * 100
explained_variance
```

Choix du nombre d'axes :

[ ]:
```
# Analyse des valeurs propres
n_components = len(pca.explained_variance_)
comp = pd.DataFrame(
    {
        "Dimension" : ["Dim" + str(x + 1) for x in range(n_components)],
        "Valeur propre" : pca.explained_variance_,
        "% variance expliquée" : np.round(pca.explained_variance_ratio_ * 100),
        "% cum. var. expliquée" : np.round(np.cumsum(pca.
 ↪explained_variance_ratio_) * 100)
    },
    columns = ["Dimension", "Valeur propre", "% variance expliquée", "% cum.␣
 ↪var. expliquée"]
)
comp
```

[ ]:
```
# Scree plot pour choisir le nombre de composantes principales
g_comp = sns.barplot(x = "Dimension",
                y = "% variance expliquée",
                palette = ["lightseagreen"],
                data = comp)
g_comp.set(ylabel = "Variance expliquée (%)")
g_comp.figure.suptitle("Variance expliquée par dimension")
plt.axhline(y = 25, linewidth = .5, color = "dimgray", linestyle = "--") # 25 =␣
 ↪100 / 4 (nb dimensions)
plt.text(3.25, 26, "25%")
```

[ ]:
```
# Calcul du cosinus carré des variables
cos_squared = np.square(pca.components_)
df_cos_squared = pd.DataFrame(cos_squared, columns=['PC{}'.format(i+1) for i in␣
 ↪range(n_components)])
df_cos_squared.index = CovidCases_without_country.columns

print(df_cos_squared)
```

[ ]:
```
# Contribution à la formation de l'axe
components = pca.components_

n_components = X.shape[1]
feature_names=CovidCases_without_country.columns

loadings = pca.components_.T
eigenvalues = pca.explained_variance_
variable_contributions = (loadings**2) * eigenvalues

column_names = [f'PC{i+1}_contrib' for i in range(n_components)]
variable_contrib_df = pd.DataFrame(variable_contributions,␣
 ↪columns=column_names, index=feature_names)
```

```
variable_contrib_df
```

[ ]:
```
# Calculer la contribution des individus à la formation des axes
eigenvalues = pca.explained_variance_
contributions = (X_pca**2) / (X_pca.shape[0] * eigenvalues)
contrib_percent = contributions * 100

column_names = [f'PC{i+1}_contrib' for i in range(n_components)]
contrib_df = pd.DataFrame(contrib_percent, columns=column_names)

print(contrib_df)
```

[ ]:
```
# Créer le cercle de corrélation
coeff = np.transpose(pca.components_[0:2, :])
n = coeff.shape[0]
xs = np.array([1, 0])
ys = np.array([0, 1])

plt.figure(figsize=(10, 10))
for i in range(n):
    plt.arrow(0, 0, coeff[i, 0], coeff[i, 1], color='k', alpha=0.9,␣
 ↪head_width=0.02)
    plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, feature_names[i],␣
 ↪color='k', ha='center', va='center')

circle = plt.Circle((0, 0), 1, color='gray', fill=False, linestyle='--')
plt.gca().add_artist(circle)
plt.xlim(-1, 1)
plt.ylim(-1, 1)
plt.axhline(0, color='gray', linewidth=1)
plt.axvline(0, color='gray', linewidth=1)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Cercle de corrélation')
plt.show()
```

[ ]:
```
# Projection des pays
plt.figure(figsize=(12, 7))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)

for i, country in enumerate(CovidCases.iloc[:, 0]):
    plt.text(X_pca[i, 0], X_pca[i, 1], country, fontsize=9, alpha=0.7)

plt.xlabel("PC1 ({}% variance)".format(round(explained_variance[0], 1)))
plt.ylabel("PC2 ({}% variance)".format(round(explained_variance[1], 1)))
```

```
plt.title("Projection des pays sur le premier plan factoriel (sans Yemen et␣
 ↪Cyprus)")
plt.axhline(0, color="grey", linestyle="--", linewidth=0.8)
plt.axvline(0, color="grey", linestyle="--", linewidth=0.8)
plt.grid()
plt.show()
```

8.3 TEST 2 PCA avec les variables : "TotalCasesPerMillion", "TotalDeathsPerMillion", "ActiveCasesPerMillion"

[ ]:
```
# Préparation nous le PCA
CovidCases = pd.read_csv("CovidCases.csv", sep=",")
CovidCases.loc[CovidCases["Country"] == "Macao", :] = CovidCases.
 ↪loc[CovidCases["Country"] == "Macao", :].fillna(0)
CovidCases = CovidCases.drop(columns=["ID"])
CovidCases['ActiveCasesPerMillion'] = CovidCases['ActiveCases'] * 1000000 /␣
 ↪CovidCases['TotalPopulation']
colums_stand = CovidCases.columns[1:]
print(colums_stand)
```

- Standardiser les données

[ ]:
```
scaler = StandardScaler()
CovidCases[colums_stand] = scaler.fit_transform(CovidCases[colums_stand])
CovidCases.head()
```

- Selection des variables

[ ]:
```
CovidCases_without_country = CovidCases[["TotalCasesPerMillion",␣
 ↪"TotalDeathsPerMillion", "ActiveCasesPerMillion"]]
CovidCases_without_country.head()
```

[ ]:
```
X = CovidCases_without_country.iloc[:, :].values
```

[ ]:
```
X
```

- PCA :

[ ]:
```
pca = PCA()
X_pca=pca.fit_transform(X)
```

[ ]:
```
# Inertie expliquée
explained_variance = pca.explained_variance_ratio_ * 100
explained_variance
```

Choix du nombre d'axes :

[ ]:
```
# Analyse des valeurs propres
n_components = len(pca.explained_variance_)
comp = pd.DataFrame(
    {
        "Dimension" : ["Dim" + str(x + 1) for x in range(n_components)],
        "Valeur propre" : pca.explained_variance_,
        "% variance expliquée" : np.round(pca.explained_variance_ratio_ * 100),
        "% cum. var. expliquée" : np.round(np.cumsum(pca.
 ↪explained_variance_ratio_) * 100)
    },
    columns = ["Dimension", "Valeur propre", "% variance expliquée", "% cum.␣
 ↪var. expliquée"]
)
comp
```

[ ]:
```
# Scree plot pour choisir le nombre de composantes principales
g_comp = sns.barplot(x = "Dimension",
                y = "% variance expliquée",
                palette = ["lightseagreen"],
                data = comp)
g_comp.set(ylabel = "Variance expliquée (%)")
g_comp.figure.suptitle("Variance expliquée par dimension")
plt.axhline(y = 25, linewidth = .5, color = "dimgray", linestyle = "--") # 25 =␣
 ↪100 / 4 (nb dimensions)
plt.text(3.25, 26, "25%")
```

[ ]:
```
# Calcul du cosinus carré des variables
cos_squared = np.square(pca.components_)
df_cos_squared = pd.DataFrame(cos_squared, columns=['PC{}'.format(i+1) for i in␣
 ↪range(n_components)])
df_cos_squared.index = CovidCases_without_country.columns

print(df_cos_squared)
```

[ ]:
```
# Contribution à la formation de l'axe
components = pca.components_

n_components = X.shape[1]
feature_names=CovidCases_without_country.columns

loadings = pca.components_.T
eigenvalues = pca.explained_variance_
variable_contributions = (loadings**2) * eigenvalues

column_names = [f'PC{i+1}_contrib' for i in range(n_components)]
variable_contrib_df = pd.DataFrame(variable_contributions,␣
 ↪columns=column_names, index=feature_names)
```

```
variable_contrib_df
```

[ ]:
```
# Calculer la contribution des individus à la formation des axes
eigenvalues = pca.explained_variance_
contributions = (X_pca**2) / (X_pca.shape[0] * eigenvalues)
contrib_percent = contributions * 100

column_names = [f'PC{i+1}_contrib' for i in range(n_components)]
contrib_df = pd.DataFrame(contrib_percent, columns=column_names)

print(contrib_df)
```

[ ]:
```
# Créer le cercle de corrélation
coeff = np.transpose(pca.components_[0:2, :])
n = coeff.shape[0]
xs = np.array([1, 0])
ys = np.array([0, 1])

plt.figure(figsize=(10, 10))
for i in range(n):
    plt.arrow(0, 0, coeff[i, 0], coeff[i, 1], color='k', alpha=0.9,␣
 ↪head_width=0.02)
    plt.text(coeff[i, 0] * 1.15, coeff[i, 1] * 1.15, feature_names[i],␣
 ↪color='k', ha='center', va='center')

circle = plt.Circle((0, 0), 1, color='gray', fill=False, linestyle='--')
plt.gca().add_artist(circle)
plt.xlim(-1, 1)
plt.ylim(-1, 1)
plt.axhline(0, color='gray', linewidth=1)
plt.axvline(0, color='gray', linewidth=1)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Cercle de corrélation')
plt.show()
```

[ ]:
```
# Projection des pays
plt.figure(figsize=(12, 7))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)

for i, country in enumerate(CovidCases.iloc[:, 0]):
    plt.text(X_pca[i, 0], X_pca[i, 1], country, fontsize=9, alpha=0.7)

plt.xlabel("PC1 ({}% variance)".format(round(explained_variance[0], 1)))
plt.ylabel("PC2 ({}% variance)".format(round(explained_variance[1], 1)))
plt.title("Projection des pays sur le premier plan factoriel")
```

```
plt.axhline(0, color="grey", linestyle="--", linewidth=0.8)
plt.axvline(0, color="grey", linestyle="--", linewidth=0.8)
plt.grid()
plt.show()
```