# Importation des bibliothèques nécessaires

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Fonction principale pour analyser le fichier Excel

def analyze\_excel\_and\_generate\_tables(file\_path, sheet\_name=0):

try:

# Lecture du fichier Excel

data = pd.read\_excel(file\_path, header=None, sheet\_name=sheet\_name)

print(f"Fichier Excel chargé avec succès depuis la feuille : {sheet\_name} !")

# Étape 1 : Détection des scénarios

scenario\_row = data.iloc[0, 1:]

scenario\_names = scenario\_row.unique()

scenario\_names\_cleaned = [name[name.find("(") + 1 : name.find(")")].strip().lower() for name in scenario\_names]

print(f"Scénarios détectés ({len(scenario\_names\_cleaned)}) : {scenario\_names\_cleaned}")

# Étape 2 : Détection des contributions

contribution\_row = data.iloc[1, 1:]

print(f"Contributions détectées : {list(contribution\_row.unique())}")

# Étape 3 : Structuration des données

categories = data.iloc[2:, 0]

values = data.iloc[2:, 1:]

values.columns = pd.MultiIndex.from\_arrays([scenario\_row, contribution\_row])

values.index = categories

# Tableau initial

initial\_table = values.copy()

# Totaux d’impact

total\_impact\_table = values.T.groupby(level=0).sum().T

# Remplacer les noms des colonnes par les noms simplifiés des scénarios

total\_impact\_table.columns = [name[name.find("(") + 1 : name.find(")")].strip().lower() for name in total\_impact\_table.columns]

# Pourcentages relatifs

relative\_percentage\_table = total\_impact\_table.copy()

for category in relative\_percentage\_table.index:

max\_value = relative\_percentage\_table.loc[category].max()

relative\_percentage\_table.loc[category] = (

(relative\_percentage\_table.loc[category] / max\_value) \* 100

)

relative\_percentage\_table = relative\_percentage\_table.add\_suffix(" (%)")

# Fusion des deux tableaux

combined\_table = pd.concat([total\_impact\_table, relative\_percentage\_table], axis=1)

return initial\_table, combined\_table, total\_impact\_table, scenario\_names\_cleaned

except Exception as e:

print(f"Erreur lors de l'analyse du fichier Excel : {e}")

return None, None, None, None

def generate\_percentage\_table(initial\_table, total\_impact\_table):

"""

Génère un tableau identique à `initial\_table`, mais avec les valeurs en pourcentage.

:param initial\_table: DataFrame initial contenant les valeurs d'impact.

:param total\_impact\_table: DataFrame contenant les totaux d'impact pour chaque catégorie.

:return: DataFrame avec les valeurs en pourcentage.

"""

try:

# Copier la structure du tableau initial

percentage\_table = initial\_table.copy()

# Calculer les pourcentages

for category in initial\_table.index:

# Récupérer le maximum dans `total\_impact\_table` pour cette catégorie

max\_total\_impact = total\_impact\_table.loc[category].max()

# Diviser chaque valeur par ce maximum et multiplier par 100

percentage\_table.loc[category] = (initial\_table.loc[category] / max\_total\_impact) \* 100

return percentage\_table

except Exception as e:

print(f"Erreur lors de la génération du tableau en pourcentage : {e}")

return None

# Fonction pour générer un tableau par scénario

def generate\_tables\_by\_scenario(initial\_table, scenario\_names\_cleaned):

"""

Génère un tableau formaté par scénario à partir du tableau initial,

avec une colonne pour le total d’impact et les pourcentages de contribution.

:param initial\_table: DataFrame initial contenant les catégories d'impact et les contributions.

:param scenario\_names\_cleaned: Liste des noms des scénarios simplifiés.

:return: Dictionnaire contenant un tableau par scénario.

"""

try:

scenario\_tables = {}

for scenario\_clean in scenario\_names\_cleaned:

matching\_scenario = [

col for col in initial\_table.columns.levels[0] if scenario\_clean in col.lower()

]

if not matching\_scenario:

raise KeyError(f"Le scénario simplifié '{scenario\_clean}' n'a pas de correspondance.")

matching\_scenario = matching\_scenario[0]

# Filtrer les données pour ce scénario

scenario\_data = initial\_table.xs(key=matching\_scenario, axis=1, level=0)

# Créer un tableau formaté

scenario\_table = pd.DataFrame()

scenario\_table["Catégorie d'impact"] = initial\_table.index

for contribution in scenario\_data.columns:

scenario\_table[contribution] = scenario\_data[contribution].values

# Calcul du total d’impact pour chaque catégorie

scenario\_table["Total d'impact"] = scenario\_table.iloc[:, 1:].sum(axis=1)

# Calcul des pourcentages de contribution

for contribution in scenario\_data.columns:

scenario\_table[f"% {contribution}"] = (

scenario\_table[contribution] / scenario\_table["Total d'impact"] \* 100

)

# Ajouter le tableau au dictionnaire

scenario\_tables[scenario\_clean] = scenario\_table

# Afficher un aperçu du tableau

print(f"\nTableau généré pour {scenario\_clean} :")

print(scenario\_table.head())

return scenario\_tables

except Exception as e:

print(f"Erreur lors de la génération des tableaux par scénario : {e}")

return None

# Fonction pour tracer l'histogramme empilé 100 %

def plot\_stacked\_bar\_chart(total\_impact\_table):

try:

normalized\_data = total\_impact\_table.div(total\_impact\_table.sum(axis=1), axis=0) \* 100

categories = normalized\_data.index

scenarios = normalized\_data.columns

bar\_width = 0.8

fig, ax = plt.subplots(figsize=(12, 7))

bottom\_stack = np.zeros(len(categories))

for scenario in scenarios:

ax.bar(

categories,

normalized\_data[scenario],

bar\_width,

label=scenario,

bottom=bottom\_stack,

)

bottom\_stack += normalized\_data[scenario]

ax.set\_title("Comparison of scenarios", fontsize=14)

ax.set\_xlabel("Impact category", fontsize=12)

ax.set\_ylabel("Contributions (%)", fontsize=12)

ax.set\_ylim(0, 100)

ax.set\_xlim(-0.5, len(categories) - 0.5)

ax.set\_xticks(range(len(categories)))

ax.set\_xticklabels(categories, rotation=45, ha="right", fontsize=10)

ax.legend(title="Scenarios", fontsize=10, bbox\_to\_anchor=(1.05, 1), loc='upper left')

ax.grid(axis="y", linestyle="--", alpha=0.7)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Erreur lors de la génération du graphique : {e}")

# Fonction pour tracer un comparaison bar chart

def plot\_comparison\_bar\_chart(total\_impact\_table):

try:

normalized\_data = total\_impact\_table.copy()

for category in normalized\_data.index:

max\_value = normalized\_data.loc[category].max()

normalized\_data.loc[category] = (normalized\_data.loc[category] / max\_value) \* 100

categories = normalized\_data.index

scenarios = normalized\_data.columns

bar\_width = 0.2

x\_positions = np.arange(len(categories))

fig, ax = plt.subplots(figsize=(12, 7))

for i, scenario in enumerate(scenarios):

ax.bar(

x\_positions + i \* bar\_width,

normalized\_data[scenario],

bar\_width,

label=scenario,

)

ax.set\_title("Comparison of scenarios", fontsize=14, pad=30)

ax.set\_xlabel("Impact category", fontsize=12)

ax.set\_ylabel("Relative percentage (%)", fontsize=12)

ax.set\_ylim(0, 100)

ax.set\_xlim(-0.5, len(categories) - 0.5)

ax.set\_xticks(x\_positions + (len(scenarios) - 1) \* bar\_width / 2)

ax.set\_xticklabels(categories, rotation=45, ha="right", fontsize=10)

ax.legend(title="Scenarios", fontsize=10, bbox\_to\_anchor=(1.05, 1), loc='upper left')

ax.grid(axis="y", linestyle="--", alpha=0.7)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Erreur lors de la génération du graphique : {e}")

# Fonction pour énérer des graphiques empilés en pourcentage pour chaque scénario

def plot\_relative\_contribution\_by\_scenario(scenario\_tables):

"""

Génère un graphique empilé en pourcentage pour chaque scénario,

montrant les contributions relatives par catégorie d'impact avec des couleurs cohérentes pour les contributions communes.

"""

try:

# Définir un dictionnaire de couleurs pour les contributions

contribution\_colors = {}

available\_colors = plt.cm.tab10.colors # Palette de couleurs prédéfinie

color\_index = 0

# Identifier toutes les contributions uniques dans tous les scénarios

all\_contributions = set()

for table in scenario\_tables.values():

contribution\_columns = [

col for col in table.columns if not col.startswith("%") and col != "Catégorie d'impact" and col != "Total d'impact"

]

all\_contributions.update(contribution\_columns)

# Attribuer une couleur à chaque contribution unique

for contribution in sorted(all\_contributions):

contribution\_colors[contribution] = available\_colors[color\_index % len(available\_colors)]

color\_index += 1

# Générer les graphiques pour chaque scénario

for scenario, table in scenario\_tables.items():

# Normaliser les contributions pour chaque catégorie d'impact

contribution\_columns = [

col for col in table.columns if not col.startswith("%") and col != "Catégorie d'impact" and col != "Total d'impact"

]

normalized\_data = table[contribution\_columns].div(table["Total d'impact"], axis=0) \* 100

# Paramètres du graphique

categories = table["Catégorie d'impact"]

bar\_width = 0.8

# Création du graphique

fig, ax = plt.subplots(figsize=(12, 7))

bottom\_stack = np.zeros(len(categories))

# Ajouter les segments pour chaque contribution

for contribution in contribution\_columns:

ax.bar(

categories,

normalized\_data[contribution],

bar\_width,

label=contribution,

bottom=bottom\_stack,

color=contribution\_colors[contribution], # Utiliser la couleur cohérente

)

bottom\_stack += normalized\_data[contribution]

# Ajouter les totaux au-dessus des barres

for i, total in enumerate(table["Total d'impact"]):

ax.text(

i, # Position x

102, # Position y (juste au-dessus de 100 %, sous le titre)

f"{total:.2f}", # Format du texte

ha="center", # Alignement horizontal

va="bottom", # Alignement vertical

fontsize=10, # Taille du texte

color="black", # Couleur du texte

)

# Personnalisation du graphique

ax.set\_title(f"Relative contributions for the scenario : {scenario.capitalize()}", fontsize=14, pad=40)

ax.set\_xlabel("Impact category", fontsize=12)

ax.set\_ylabel("Contribution (%)", fontsize=12)

ax.set\_ylim(0, 100) # Ajout d'une marge au-dessus de 100 %

ax.set\_xlim(-0.5, len(categories) - 0.5)

ax.set\_xticks(range(len(categories)))

ax.set\_xticklabels(categories, rotation=45, ha="right", fontsize=10)

ax.legend(title="Contributions", fontsize=10, bbox\_to\_anchor=(1.05, 1), loc='upper left')

ax.grid(axis="y", linestyle="--", alpha=0.7)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Erreur lors de la génération des graphiques de contribution relative : {e}")

# Fonction pour énérer des graphiques empilés horizontales en pourcentage pour chaque scénario

def plot\_relative\_contribution\_by\_scenario\_horizontal(scenario\_tables):

"""

Génère un graphique empilé en pourcentage horizontal pour chaque scénario,

montrant les contributions relatives par catégorie d'impact avec des couleurs cohérentes pour les contributions communes.

"""

try:

# Définir un dictionnaire de couleurs pour les contributions

contribution\_colors = {}

available\_colors = plt.cm.tab10.colors # Palette de couleurs prédéfinie

color\_index = 0

# Identifier toutes les contributions uniques dans tous les scénarios

all\_contributions = set()

for table in scenario\_tables.values():

contribution\_columns = [

col for col in table.columns if not col.startswith("%") and col != "Catégorie d'impact" and col != "Total d'impact"

]

all\_contributions.update(contribution\_columns)

# Attribuer une couleur à chaque contribution unique

for contribution in sorted(all\_contributions):

contribution\_colors[contribution] = available\_colors[color\_index % len(available\_colors)]

color\_index += 1

# Générer les graphiques pour chaque scénario

for scenario, table in scenario\_tables.items():

# Normaliser les contributions pour chaque catégorie d'impact

contribution\_columns = [

col for col in table.columns if not col.startswith("%") and col != "Catégorie d'impact" and col != "Total d'impact"

]

normalized\_data = table[contribution\_columns].div(table["Total d'impact"], axis=0) \* 100

# Paramètres du graphique

categories = table["Catégorie d'impact"]

bar\_height = 0.8

# Création du graphique

fig, ax = plt.subplots(figsize=(12, 7))

left\_stack = np.zeros(len(categories))

# Ajouter les segments pour chaque contribution

for contribution in contribution\_columns:

ax.barh(

categories,

normalized\_data[contribution],

bar\_height,

label=contribution,

left=left\_stack,

color=contribution\_colors[contribution], # Utiliser la couleur cohérente

)

left\_stack += normalized\_data[contribution]

# Ajouter les totaux à droite des barres, avec un léger décalage

for i, total in enumerate(table["Total d'impact"]):

ax.text(

105, # Position x (légèrement à droite de 100 %)

i, # Position y

f"{total:.2f}", # Format du texte

va="center", # Alignement vertical

ha="left", # Alignement horizontal

fontsize=10, # Taille du texte

color="black", # Couleur du texte

)

# Personnalisation du graphique

ax.set\_title(f"Relative contributions for the scenario : {scenario.capitalize()}", fontsize=14, pad=20)

ax.set\_xlabel("Contribution (%)", fontsize=12)

ax.set\_ylabel("Impact category", fontsize=12)

ax.set\_xlim(0, 110) # Ajout d'une marge à droite de 100 %

ax.set\_ylim(-0.5, len(categories) - 0.5)

ax.set\_yticks(range(len(categories)))

ax.set\_yticklabels(categories, fontsize=10)

ax.legend(title="Contributions", fontsize=10, bbox\_to\_anchor=(1.05, 1), loc='upper left')

ax.grid(axis="x", linestyle="--", alpha=0.7)

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Erreur lors de la génération des graphiques horizontaux de contribution relative : {e}")

def plot\_stacked\_bar\_by\_category(percentage\_table, total\_impact\_table):

"""

Génère un graphique empilé pour chaque catégorie d'impact.

Chaque graphique compare les contributions de chaque scénario pour une catégorie donnée.

Les totaux des scénarios sont affichés au-dessus de chaque barre.

:param percentage\_table: DataFrame contenant les valeurs en pourcentage, avec un MultiIndex (scénarios, contributions).

:param total\_impact\_table: DataFrame contenant les valeurs totales d'impact pour chaque catégorie.

"""

try:

# Récupérer les catégories d'impact

categories = percentage\_table.index

scenarios = percentage\_table.columns.levels[0] # Scénarios

contributions = percentage\_table.columns.levels[1] # Contributions

# Extraire uniquement les noms simplifiés des scénarios

scenario\_labels = [scenario.split("(")[-1].replace(")", "").strip() for scenario in scenarios]

# Générer un graphique pour chaque catégorie

for category in categories:

# Extraire les données pour cette catégorie

category\_data = percentage\_table.loc[category]

# Préparer les positions des barres pour les scénarios

x\_positions = np.arange(len(scenarios))

bar\_width = 0.6 # Largeur des barres

# Initialiser la figure

fig, ax = plt.subplots(figsize=(12, 7))

# Initialiser les bases pour empiler les contributions

bottom\_stack = np.zeros(len(scenarios))

# Parcourir les contributions pour empiler les données

for contribution in contributions:

contribution\_values = [

category\_data[(scenario, contribution)] if (scenario, contribution) in category\_data.index else 0

for scenario in scenarios

]

# Ajouter une barre empilée pour cette contribution

ax.bar(

x\_positions,

contribution\_values,

width=bar\_width,

label=contribution,

bottom=bottom\_stack

)

# Mettre à jour la pile

bottom\_stack += contribution\_values

# Ajouter les valeurs totales au-dessus des barres

for i, scenario in enumerate(scenarios):

total\_value = total\_impact\_table.loc[category, scenario\_labels[i]]

ax.text(

x\_positions[i], # Position x

bottom\_stack[i] + 2, # Position y légèrement au-dessus de la barre

f"{total\_value:.2f}", # Format du texte

ha="center", # Alignement horizontal

va="bottom", # Alignement vertical

fontsize=10, # Taille du texte

color="black", # Couleur du texte

)

# Personnalisation des axes et du graphique

ax.set\_title(f"Analysis of contributions - Category : {category}", fontsize=16, pad=30)

ax.set\_xlabel("Scenarios", fontsize=14)

ax.set\_ylabel("Contribution (%)", fontsize=14)

ax.set\_xticks(x\_positions)

ax.set\_xticklabels(scenario\_labels, rotation=45, ha="right", fontsize=12)

ax.set\_ylim(0, 100) # Ajuster pour laisser de la place aux annotations

ax.legend(title="Contributions", fontsize=10, bbox\_to\_anchor=(1.05, 1), loc="upper left")

ax.grid(axis="y", linestyle="--", alpha=0.7)

# Ajuster les marges et afficher

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Erreur lors de la génération des graphiques : {e}")

def plot\_combined\_graph\_with\_scenario\_hatches(percentage\_table, total\_impact\_table):

"""

Génère un graphique combiné avec toutes les catégories sur un seul graphique.

Les barres sont colorées par contribution et les scénarios sont identifiés par des motifs variés.

La légende combinée affiche les contributions (couleurs) et les scénarios (motifs).

Les valeurs totales de l'impact sont affichées au-dessus de chaque barre.

"""

try:

# Récupérer les catégories, scénarios et contributions

categories = percentage\_table.index

scenarios = percentage\_table.columns.levels[0] # Scénarios

contributions = percentage\_table.columns.levels[1] # Contributions

# Préparer les positions des groupes de barres (chaque groupe = une catégorie)

x\_positions = np.arange(len(categories)) # Une position pour chaque catégorie

bar\_width = 0.9 / len(scenarios) # Réduire la largeur des barres pour plusieurs scénarios

total\_width = bar\_width \* len(scenarios) # Largeur totale d'un groupe

# Définir un dictionnaire de couleurs pour les contributions

contribution\_colors = {}

available\_colors = plt.cm.tab10.colors # Palette de couleurs

for i, contribution in enumerate(contributions):

contribution\_colors[contribution] = available\_colors[i % len(available\_colors)]

# Définir des motifs (hatch patterns) pour chaque scénario

scenario\_hatches = ["//", "xx", "..", "oo", "--"] # Motifs variés

scenario\_hatch\_dict = {scenario: scenario\_hatches[i % len(scenario\_hatches)] for i, scenario in enumerate(scenarios)}

# Extraire uniquement les noms des scénarios (par exemple, lab, upscale, hybrid)

scenario\_labels = [scenario.split("(")[-1].replace(")", "").strip() for scenario in scenarios]

# Initialiser la figure

fig, ax = plt.subplots(figsize=(15, 8))

# Parcourir chaque scénario pour tracer les groupes de barres

for i, scenario in enumerate(scenarios):

# Initialiser la base pour empiler les contributions

bottom\_stack = np.zeros(len(categories))

# Ajouter les contributions empilées pour ce scénario

for contribution in contributions:

contribution\_values = [

percentage\_table.loc[category, (scenario, contribution)] if (scenario, contribution) in percentage\_table.columns else 0

for category in categories

]

# Ajouter les barres empilées pour cette contribution

ax.bar(

x\_positions + i \* bar\_width - total\_width / 2, # Décalage des groupes de barres

contribution\_values,

width=bar\_width,

color=contribution\_colors[contribution], # Couleur par contribution

bottom=bottom\_stack,

hatch=scenario\_hatch\_dict[scenario] # Motif par scénario

)

# Mettre à jour la pile pour le prochain niveau

bottom\_stack += contribution\_values

# Ajouter les valeurs totales au-dessus des barres

for j, category in enumerate(categories):

total\_value = total\_impact\_table.loc[category, scenario\_labels[i]]

ax.text(

x\_positions[j] + i \* bar\_width - total\_width / 2, # Position x

bottom\_stack[j] + 2, # Position y (légèrement au-dessus de la barre)

f"{total\_value:.2f}", # Format du texte

ha="center", # Alignement horizontal

va="bottom", # Alignement vertical

fontsize=8, # Taille du texte

color="black", # Couleur du texte

)

# Configuration des axes et du graphique

ax.set\_title("Combined comparison of contributions and scenarios", fontsize=16, pad=30)

ax.set\_xlabel("Impact category", fontsize=14)

ax.set\_ylabel("Contribution (%)", fontsize=14)

ax.set\_xticks(x\_positions)

ax.set\_xticklabels(categories, rotation=45, ha="right", fontsize=12)

ax.set\_ylim(0, 100) # Limiter l'axe Y à 110 % pour laisser de la place aux annotations

ax.grid(axis="y", linestyle="--", alpha=0.7)

# Légende combinée : Contributions et Scénarios

legend\_handles = []

legend\_labels = []

# Ajouter les contributions à la légende

for contribution, color in contribution\_colors.items():

legend\_handles.append(plt.Line2D([0], [0], color=color, lw=10))

legend\_labels.append(contribution)

# Ajouter les scénarios à la légende avec des rectangles remplis de motifs

for scenario, hatch in zip(scenario\_labels, scenario\_hatch\_dict.values()):

legend\_handles.append(

plt.Rectangle((0, 0), 1, 1, facecolor="white", edgecolor="black", hatch=hatch, lw=1.5)

)

legend\_labels.append(scenario) # Utiliser le nom simplifié du scénario

# Afficher la légende combinée

ax.legend(

legend\_handles,

legend\_labels,

title="Contributions and Scenarios",

fontsize=10,

loc="upper left",

bbox\_to\_anchor=(1.05, 1)

)

# Ajuster les marges et afficher

plt.tight\_layout()

plt.show()

except Exception as e:

print(f"Erreur lors de la génération du graphique combiné avec motifs : {e}")

# Exemple d'utilisation

if \_\_name\_\_ == "\_\_main\_\_":

file\_path = "C:\\Users\idriouech001\Desktop\BIOCOP Project\Code contribution analysis\données.xlsx"

sheet\_name = "Données"

# Analyse du fichier Excel et génération des tableaux

initial\_table, combined\_table, total\_impact\_table, scenario\_names\_cleaned = analyze\_excel\_and\_generate\_tables(

file\_path, sheet\_name=sheet\_name

)

if initial\_table is not None and combined\_table is not None:

print("\nLes deux tableaux principaux ont été générés avec succès.")

# Génération des tableaux par scénario

scenario\_tables = generate\_tables\_by\_scenario(initial\_table, scenario\_names\_cleaned)

# Génération du tableau en pourcentage

percentage\_table = generate\_percentage\_table(initial\_table, total\_impact\_table)

if percentage\_table is not None:

print("\nTableau en pourcentage généré avec succès :")

print(percentage\_table.head())

# Sauvegarde du tableau en pourcentage

percentage\_table.to\_excel("tableau\_en\_pourcentage.xlsx", index=True)

# Génération du graphique combiné pour toutes les catégories avec motifs pour les scénarios

print("\nGénération du graphique combiné avec motifs pour les scénarios...")

plot\_combined\_graph\_with\_scenario\_hatches(percentage\_table, total\_impact\_table)

# Sauvegarde des tableaux par scénario

for scenario, table in scenario\_tables.items():

table.to\_excel(f"{scenario}\_tableau.xlsx", index=False)

# Génération des graphiques empilés en pourcentage par scénario

plot\_relative\_contribution\_by\_scenario(scenario\_tables)

# Génération de l'histogramme empilé 100 %

plot\_stacked\_bar\_chart(total\_impact\_table)

# Génération de l'histogramme côte à côte (référentiel 100 %)

plot\_comparison\_bar\_chart(total\_impact\_table)

# Génération des graphiques empilés en pourcentage horizontaux par scénario

plot\_relative\_contribution\_by\_scenario\_horizontal(scenario\_tables)

# Génération des graphiques par catégorie

plot\_stacked\_bar\_by\_category(percentage\_table, total\_impact\_table)