import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.colors as mcolors

import numpy as np

from dateutil.parser import parse

import statsmodels.api as sm

# Charger les données depuis le fichier Excel

data = pd.read\_excel("C:\\Monnaie\\Eviews\\USA\\Variables\_USA.xlsx", parse\_dates=True)

data['Date'] = pd.to\_datetime(data['Date'], errors='coerce')

data['Date'] = pd.to\_datetime(data['Date'], format='%Y-%m-%d')

# Ajouter une nouvelle colonne 'Date\_parsed' au DataFrame

data['Date\_parsed'] = pd.to\_datetime(data['Date'], infer\_datetime\_format=True)

data.set\_index('Date\_parsed', inplace=True)

start\_date\_period = data.index.min().strftime("%b %Y")

end\_date\_period = data.index.max().strftime('%b %Y')

coef\_list\_simple = []

coef\_list\_simple\_cumul = []

# La taille de la fenêtre, i.e. le nombre de périodes (mois, trimestres ou années

#w = input("Entrez, en mois, la taille de la fenêtre : ")

#w = int(w)

w = 24

# Initialiser les variables

variable\_1 = 'M2'

variable\_2 = 'cpi'

# Boucle à travers les graphiques

for i in range(len(data)-w+1):

# Extraire les données pour la fenêtre actuelle

window = data.iloc[i:i+w].copy()

fig, ax = plt.subplots(figsize=(9.7,6))

# Tracer tous les points en gris clair en arrière-plan

ax.scatter(data[variable\_1], data[variable\_2], color='lightgray', linewidth=0)

# Afficher les statistiques de la régression

from scipy import stats

x = window[variable\_1]

y = window[variable\_2]

# Ajuster les dates de début et de fin de la période

if isinstance(window.index, pd.DatetimeIndex):

start\_date = window.index.min().strftime("%b %Y")

end\_date = window.index.max().strftime('%b %Y')

else:

window.index = pd.to\_datetime(df.index, format='%Y-%m-%d')

start\_date = window.index.min().strftime("%b-%Y")

end\_date = window.index.max().strftime(" %b-%Y")

slope, tvalue, r\_value, p\_value, std\_err = stats.linregress(x, y)

r\_squared = r\_value\*\*2

tvalue = slope / std\_err

ax.text(0.95, 0.30, f"R-carré: {r\_squared:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.25, f"Coefficient: {slope:.4f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.20, f"Erreur standard : {std\_err:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.15, f"t-statistique: {tvalue:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.10, f"Probabilité: {p\_value:.2f}", transform=ax.transAxes, ha='right', va='top')

coef\_list\_simple.append({slope})

#print(coef\_list\_simple)

#coef\_list\_simple = np.array([coef\_list\_simple])

#coef\_list\_simple\_cumul.extend(coef\_list\_simple)

ax.scatter(window.loc[:, variable\_1], window.loc[:, variable\_2], color='#696969', linewidth=0)

ax.scatter(window.loc[window.index.isin(range(i, i+w)), variable\_1], window.loc[window.index.isin(range(i, i+w)), variable\_2], color='coral')

ax.plot(window.loc[window.index.isin(range(i, i+w)), variable\_1], window.loc[window.index.isin(range(i, i+w)), variable\_2], color='coral', linestyle='--')

ax.set\_xlabel("M2 (taux de variation d'une année sur l'autre)")

ax.set\_ylabel("Inflation (taux de variation d'une année sur l'autre)")

ax.set\_title("USA - M2 vs Inflation - Fenêtre de 2 ans")

ax.set\_xlim([data[variable\_1].min(), data[variable\_1].max()])

ax.set\_ylim([data[variable\_2].min(), data[variable\_2].max()])

# ax.text(0.05, 0.95, f"{window.index[0].strftime('%b %Y')} - {window.index[-1].strftime('%b %Y')}", transform=ax.transAxes, ha='left', va='top')

ax.text(0.10, 0.95, f"Période : de {start\_date\_period} à {end\_date\_period}", transform=ax.transAxes)

ax.text(0.50, 0.95, f"Fenêtre : de {start\_date} à {end\_date}", transform=ax.transAxes)

# Condition pour extraire les dates de début et de fin de période

if isinstance(window.index[1], pd.Timestamp) and isinstance(window.index[-1], pd.Timestamp):

# Afficher la droite de régression

x = window[variable\_1]

y = window[variable\_2]

coeffs = np.polyfit(x, y, 1)

reg\_line = f"y = {coeffs[0]:.2f}x + {coeffs[1]:.2f}"

ax.plot(x, np.polyval(coeffs, x), 'r', label=reg\_line)

start\_date = window.index[0]

end\_date = window.index[-1]

date\_filter = (data.index >= start\_date) & (data.index <= end\_date)

filtered\_data = data.loc[date\_filter]

# Pour USA

# ax.text(0.5, -0.12, "Créé par @robert\_cauneau - Source : U.S. Bureau of Labor Statistics, FRED, Federal Reserve Bank of St. Louis",

# Pour France, Zone Euro, NZ

ax.text(0.5, -0.12, "Créé par @robert\_cauneau - Source : FRED",

transform=ax.transAxes, ha='center', va='center', fontsize=8)

#ax.text(0.5, -0.12, "Créé par @robert\_cauneau - Source : Banque Mondiale",

# #transform=ax.transAxes, ha='center', va='center', fontsize=8)

#transform=ax.transAxes, ha='center', va='center', fontsize=8)

# définition de la marge intérieure

margin = 0.15

xmin, xmax = ax.get\_xlim()

ymin, ymax = ax.get\_ylim()

xmargin = (xmax - xmin) \* margin

ymargin = (ymax - ymin) \* margin

# ajustement des limites de l'axe avec la marge intérieure

ax.set\_xlim(xmin - xmargin, xmax + xmargin)

ax.set\_ylim(ymin - ymargin, ymax + ymargin)

plt.savefig(f"C:\\temp\\w{i:03d}.png", dpi=100)

plt.show()

plt.close()

# Impression des résultats de la régression sur toute la période

# Ajout d'une constante à la variable d'entrée x

#x = sm.add\_constant(x)

# Création du modèle de régression linéaire

#model = sm.OLS(y, x).fit()

# Séparation des variables dépendantes (y) et indépendantes (x)

x = data['M2']

y = data['cpi']

# Création du modèle de régression linéaire

model = sm.OLS(y, sm.add\_constant(x)).fit()

# Extraction des résultats

coefficients = model.params # Coefficients estimés

std\_errors = model.bse # Erreurs standard des coefficients

r\_squared = model.rsquared # Coefficient de détermination R²

p\_values = model.pvalues # Valeurs-p associées aux coefficients

t\_values = model.tvalues # Valeurs-p associées aux coefficients

fig, ax = plt.subplots(figsize=(9.7,6))

ax.text(0.5, 0.95, f"Régression pour toute la période : de {start\_date\_period} à {end\_date\_period}",

transform=ax.transAxes, ha='center', va='center', fontsize=14)

ax.text(0.95, 0.30, f"R-carré: {r\_squared:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.25, f"Coefficient: {coefficients[1]:.4f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.20, f"Erreur standard : {std\_errors[1]:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.15, f"t-statistique: {t\_values[1]:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.95, 0.10, f"Probabilité: {p\_values[1]:.2f}", transform=ax.transAxes, ha='right', va='top')

ax.text(0.5, -0.12, "Créé par @robert\_cauneau - Source : FRED",

transform=ax.transAxes, ha='center', va='center', fontsize=8)

ax.scatter(data.loc[:, variable\_1], data.loc[:, variable\_2], color='#696969', linewidth=0)

# Afficher la droite de régression

coeffs = np.polyfit(x, y, 1)

reg\_line = f"y = {coeffs[0]:.2f}x + {coeffs[1]:.2f}"

ax.plot(x, np.polyval(coeffs, x), 'r', label=reg\_line)

ax.set\_xlabel("M2 (taux de variation d'une année sur l'autre)")

ax.set\_ylabel("Inflation (taux de variation d'une année sur l'autre)")

ax.set\_title("USA - M2 vs Inflation")

ax.set\_ylim([data[variable\_2].min(), data[variable\_2].max()])

margin = 0.15

xmin, xmax = ax.get\_xlim()

ymin, ymax = ax.get\_ylim()

xmargin = (xmax - xmin) \* margin

ymargin = (ymax - ymin) \* margin

# ajustement des limites de l'axe avec la marge intérieure

ax.set\_xlim(xmin - xmargin, xmax + xmargin)

ax.set\_ylim(ymin - ymargin, ymax + ymargin)

plt.savefig(f"C:\\temp\\w{i:03d}.png", dpi=100)

plt.show()

plt.close()

# Créer la liste des noms de fichiers png dans le dossier temp

import os

# Générer la vidéo

import imageio

# Obtenez la liste de tous les fichiers png dans le dossier temp

png\_files = [os.path.join('c:\\temp', f) for f in os.listdir('c:\\temp') if f.endswith('.png')]

# Créez la liste d'images à partir des fichiers png

images = []

for png\_file in png\_files:

images.append(imageio.imread(png\_file))

# Spécifiez le chemin complet du fichier de destination

destination\_file = r'C:\\temp\\Vidéos\\revolving-phillips.gif'

# Enregistrez les images dans le fichier de destination

#imageio.mimsave(destination\_file, images, format='GIF')

imageio.mimsave(destination\_file, images, format='GIF', duration=0.15)

print("Vidéo créée avec succès !")

#Supprimer les fichiers png

for png\_file in png\_files:

os.remove(png\_file)