MCL732 - ASSIGNMENT 1 (PM10)

by Abhisar 2021ME11019

Creating :

- weighting matrix (W)
- Source Matrix (A)
- Specie Contribution Matrix (C)

```
import pandas as pd
import numpy as np

def create_weighting_matrix(excel_path):
    # Read the Excel file
    df = pd.read_excel(excel_path)

    # Extract the standard deviation values from the 'SD' column
    standard_deviations = df['SD'].values

    # Create the diagonal matrix with entries as 1/(standard deviation)^2
    weighting_matrix = np.diag(1 / standard_deviations**2)

    return weighting_matrix

def create_source_matrix(excel_path):
    # Read the Excel file
    df = pd.read_excel(excel_path, index_col=0)  # Assuming species names are in the first column

    # Extract numerical values associated with different sources
    source_matrix = df.values

    return source_matrix

def create_vector_c(excel_path):
    # Read the Excel file
    df = pd.read_excel(excel_path)

    # Extract values from the second column
    vector_c = df.iloc[0:, 1].values

    # Convert the vector to a column vector
    vector_c = vector_c.reshape(-1, 1)

    return vector_c

def cmb_solution(A, W, C):
    # Calculate the solution using Equation 26.16
    ATWA_inv = np.linalg.inv(A.T @ W @ A)
    s = ATWA_inv @ A.T @ W @ C

    return s

# Excel paths to access data
excel_path_W = 'SD 10.xlsx'
excel_path_A = 'Matrix A.xlsx'
excel_path_C = 'C 10.xlsx'

# Create the weighting matrix
W = create_weighting_matrix(excel_path_W)

# Create the source matrix
A = create_source_matrix(excel_path_A)

# Create the vector C
C = create_vector_c(excel_path_C)

# Apply the CMB solution
source_contributions = cmb_solution(A, W, C)
#####################################################################################################################################
```

Finding Different Sources for which data is available

```
####################################################################################################################################
def extract_source_names(excel_path):
    # Read the Excel file
    df = pd.read_excel(excel_path)
    source_names = df.columns[1:].tolist()
    return source_names

excel_path = 'Matrix A.xlsx'

source_names = extract_source_names(excel_path)
####################################################################################################################################
```

Calculationg Source Profiles for Matrix W

```
source = extract_source_names('Matrix A.xlsx')
for i in range(len(source)):
  print(source[i],":",source_contributions[i][0] * 100)
```

```
Paved Road Dust : 10.71546368208154
Vegetative Burning : 19.201460200704684
Primary Crude Oil : 0.5474094847386829
Motor Vehicles : 5.8912606564048495
Limestone : 2.1214197885041126
MARINE : -0.14516932297973562
(NH4)2SO4 : 3.4708381128743366
NH4NO3 : 13.760266435437615
Secondary OC : -0.6982929845547994
NaNO3 : -0.6761553272858786
```

Creating V matrix

```
import pandas as pd
import numpy as np
##################################################################################################################################
def calculate_diagonal_matrix(eq_26_17_excel_path, source_contribution_vector, sd_vector):
    # Read the Excel file for original matrix (Oij)
    df_eq_26_17 = pd.read_excel(eq_26_17_excel_path)

    # Extract data starting from the second column
    data = df_eq_26_17.iloc[:, 1:].values

    # Transpose source_contribution_vector to make it a row vector
    source_contribution_vector_row = source_contribution_vector.T

    # Calculate the sum of squares part
    sum_of_squares_part = np.sum((source_contribution_vector_row**2) * (data**2), axis=1)

    # Calculate the diagonal matrix entries using Eq. 26.17
    diagonal_entries = 1 / (sd_vector**2 + sum_of_squares_part)

    # Create the diagonal matrix
    diagonal_matrix = np.diag(diagonal_entries)

    return diagonal_matrix

eq_26_17_excel_path = 'eq 26.17.xlsx'


source_contribution_vector = source_contributions * 100

sd_vector_excel_path = 'SD 10.xlsx'
sd_vector_df = pd.read_excel(sd_vector_excel_path)
sd_vector = sd_vector_df['SD'].values
##################################################################################################################################
```

Calculating V Matrix - Iteration 1 (V1)

```
# Calculate the diagonal matrix using Eq. 26.17
V1 = calculate_diagonal_matrix(eq_26_17_excel_path, source_contribution_vector, sd_vector)
# print("V1 matrix")
# print(V1)
```

Source Profiles as per V1 - (source_contributions2)

```
source_contributions2 = cmb_solution(A, V1, C)
source = extract_source_names('Matrix A.xlsx')
for i in range(len(source)):
  print(source[i],source_contributions2[i] * 100)
```

```
Paved Road Dust [20.99746248]
Vegetative Burning [25.64348671]
Primary Crude Oil [0.82789885]
Motor Vehicles [12.05779941]
Limestone [1.11463189]
MARINE [-0.14742566]
(NH4)2SO4 [2.96787522]
NH4NO3 [14.34603953]
Secondary OC [-8.64823028]
NaNO3 [-1.33971679]
```

Calculating V Matrix - Iteration 2 (V2)

```
V2 = calculate_diagonal_matrix(eq_26_17_excel_path, source_contributions2, sd_vector)
# print(V2)
```

Source Profiles as per V2 - (source_contributions3)

```
source_contributions3 = cmb_solution(A, V2, C)
source = extract_source_names('Matrix A.xlsx')
for i in range(len(source)):
  print(source[i],source_contributions3[i] * 100)
```

```
Paved Road Dust [10.71740177]
Vegetative Burning [19.11004137]
Primary Crude Oil [0.55338127]
Motor Vehicles [7.34529366]
Limestone [2.05442028]
MARINE [-0.11537542]
(NH4)2SO4 [3.40746423]
NH4NO3 [13.83750423]
Secondary OC [-1.38215729]
NaNO3 [-0.75768748]
```

Calculating V Matrix - Iteration 3 (V3)

```
V3 = calculate_diagonal_matrix(eq_26_17_excel_path, source_contributions3, sd_vector)
# print(V3)
```

Source Profiles as per V3 - (source_contributions4)

```
source_contributions4 = cmb_solution(A, V3, C)
source = extract_source_names('Matrix A.xlsx')
for i in range(len(source)):
  print(source[i],source_contributions4[i] * 100)
```

```
Paved Road Dust [10.59937685]
Vegetative Burning [19.27577168]
Primary Crude Oil [0.56895564]
Motor Vehicles [6.65399988]
Limestone [2.10054358]
MARINE [-0.13506998]
(NH4)2SO4 [3.4310954]
NH4NO3 [13.80819892]
Secondary OC [-1.08873864]
NaNO3 [-0.7275833]
```

Source Contribution after a given number of itertions

```
##################################################################################################################################
sc = source_contributions
for i in range(10):
  Vi = calculate_diagonal_matrix(eq_26_17_excel_path, sc, sd_vector)
  sc = cmb_solution(A, Vi, C)
  print("Iteration number : ", i+1)
  print()
  for j in range(len(source)):
    print(source[j],sc[j][0] * 100)
##################################################################################################################################

```

```
Iteration number :  1

Paved Road Dust 10.65056250801015
Vegetative Burning 19.379564244332567
Primary Crude Oil 0.5692705424830495
Motor Vehicles 6.420840308636251
Limestone 2.1053743299827796
MARINE -0.14365911020020278
(NH4)2SO4 3.4395430159226072
NH4NO3 13.797556014914344
Secondary OC -1.0288748276310027
NaNO3 -0.7169266095575135
Iteration number :  2

Paved Road Dust 10.630481708115855
Vegetative Burning 19.352808878987755
Primary Crude Oil 0.5693959720561725
Motor Vehicles 6.504035900685601
Limestone 2.1038332175263905
MARINE -0.14103417413115182
(NH4)2SO4 3.4363275620801685
NH4NO3 13.801558703525952
Secondary OC -1.0544660152416374
NaNO3 -0.7210123076180178
Iteration number :  3

Paved Road Dust 10.627199259816177
Vegetative Burning 19.346120527709147
Primary Crude Oil 0.5694273736555977
Motor Vehicles 6.51742284460285
Limestone 2.103604814805277
MARINE -0.14050935600137662
(NH4)2SO4 3.435839147747776
NH4NO3 13.802176628866
Secondary OC -1.0575109979911597
NaNO3 -0.7216268370496083
Iteration number :  4

Paved Road Dust 10.626672901301502
Vegetative Burning 19.344933644650883
```

```
Primary Crude Oil 0.569433044834204
Motor Vehicles 6.5195857935184325
Limestone 2.103567817711756
MARINE -0.14041992554996327
(NH4)2SO4 3.4357614826833327
NH4NO3 13.802275354892446
Secondary OC -1.057956378161691
NaNO3 -0.7217242708915426
Iteration number :  5

Paved Road Dust 10.62658807578533
Vegetative Burning 19.344736889768953
Primary Crude Oil 0.5694339691995949
Motor Vehicles 6.519935559942723
Limestone 2.103561811333973
MARINE -0.14040525118599173
(NH4)2SO4 3.4357489855849326
NH4NO3 13.802291262771435
```

Solution of D Part 1

```
#######################################################################################################################
# Solving D1
excel_path_W = 'SD 10.xlsx' #this matrix contains standard deviations
excel_path_A_D1 = 'D1_A Matrix.xlsx' #this contians the aij matrix
excel_path_C = 'C 10.xlsx' #this contians the total contribution of each specie
eq_26_17_excel_path = 'eq 26.17.xlsx' #this contains sigma_aij for V matrix
# Create the weighting matrix
W = create_weighting_matrix(excel_path_W)

# Create the source matrix
A_D1 = create_source_matrix(excel_path_A_D1)

# Create the vector C
C = create_vector_c(excel_path_C)

# Apply the CMB solution
source_contributions_D1 = cmb_solution(A_D1, W, C)
source = extract_source_names('D1_A Matrix.xlsx')
for i in range(len(source)):
  print(source[i],source_contributions_D1[i] * 100)
  #####################################################################################################################
```

```
Paved Road Dust [10.72729015]
Vegetative Burning [19.00827014]
Primary Crude Oil [0.5472701]
Motor Vehicles [5.85958612]
Limestone [2.12309141]
MARINE [-0.13359869]
(NH4)2SO4 [3.47426258]
NH4NO3 [13.756843]
Secondary OC [0.17598656]
NaNO3 [-0.67129155]
```

Solution of D Part 2

```
#######################################################################################################################
# Solving D2
excel_path_W = 'SD 10.xlsx' #this matrix contains standard deviations
excel_path_A_D2 = 'D2_A Matrix.xlsx' #this contians the aij matrix
excel_path_C = 'C 10.xlsx' #this contians the total contribution of each specie
eq_26_17_excel_path = 'eq 26.17.xlsx' #this contains sigma_aij for V matrix
# Create the weighting matrix
W = create_weighting_matrix(excel_path_W)

# Create the source matrix
A_D2 = create_source_matrix(excel_path_A_D2)

# Create the vector C
C = create_vector_c(excel_path_C)

# Apply the CMB solution
source_contributions_D2 = cmb_solution(A_D2, W, C)
source = extract_source_names('D2_A Matrix.xlsx')
for i in range(len(source)):
  print(source[i],source_contributions_D2[i] * 100)
#######################################################################################################################
```

```
Paved Road Dust [10.20306131]
Vegetative Burning [19.51018709]
Primary Crude Oil [0.55648503]
Motor Vehicles [6.26126554]
Limestone [0.12458349]
MARINE [-0.14668274]
(NH4)2SO4 [3.53454222]
NH4NO3 [13.6818]
Secondary OC [-0.92037426]
NaNO3 [-0.59469419]
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.