Task 2

Kosovan Ivan

In [1]:

```
import glob
import os
import pandas as pd
import numpy as np
```

Part 1

Write a function named pollutantmean that calculates the mean of a pollutant (sulfate or nitrate) across a specified list of monitors. The function pollutantmean takes three arguments: directory, pollutant, and id. Given a vector monitor ID numbers, pollutantmean reads that monitors' particulate matter data from the directory specified in the directory argument and returns the mean of the pollutant across all of the monitors, ignoring any missing values coded as NA.

In [2]:

```
def pollutantmean(dirname, pollunt, ids):
    ''' ids -> str, R format "start:stop" '''
    try:
        start, stop = ids.split(':')
        start, stop = int(start), int(stop)
    except:
        start, stop = int(ids), int(ids)

    sum_all = 0
    counter = 0
    for i in range(start, stop+1):
        if i < 10: i_str = '00' + str(i)
        if i >= 10 and int(i) < 100: i_str = '0' + str(i)
        if i >= 100: i_str = str(i)
        files = glob.glob(f'{dirname}/{i_str}.csv')
        dataset = pd.DataFrame()
        for file in files:
            data = pd.read_csv(file)
            sum_all += data[pollunt].sum()
            counter += data[pollunt].count()
    return sum_all/counter
```

In [3]:

```
pollutantmean('specdata', 'sulfate', '1:10')
```

Out[3]:

```
4.06412824256036
```

In [4]:

```
pollutantmean('specdata', 'nitrate', '70:72')
```

Out[4]:

```
1.7060473516949153
```

In [5]:

```
pollutantmean('specdata', 'nitrate', '23')
```

Out[5]:

```
1.2808333333333333
```

Part 2

Write a function named complete that reads a directory full of files and reports the number of completely observed cases in each data file. The function should return a data frame where the first column is the name of the file and the second column is the number of complete cases.

In [6]:

```
def complete(dirname, ids):
    if type(ids) is str:
        start, stop = ids.split(':')
        start, stop = int(start), int(stop)
        if start < stop:
            ns = range(start, stop+1)
        else:
            ns = range(stop, start+1)
    elif type(ids) is int: ns = [ids]
    elif type(ids) is list: ns = ids
    else:
        print("Enter data in format 'n', or 'm:n', or '[k, m, n]'")
        return

    res = pd.DataFrame(columns=['id', 'nobs'])
    for i in ns:
        if i < 10: i_str = '00' + str(i)
        if i >= 10 and int(i) < 100: i_str = '0' + str(i)
        if i >= 100: i_str = str(i)
        files = glob.glob(f'{dirname}/{i_str}.csv')
        dataset = pd.DataFrame()
        for file in files:
            data = pd.read_csv(file).dropna()
            amount = len(data.index)
            list_row = {"id":i, "nobs":amount}
            res.loc[len(res)] = list_row
    return res
```

In [7]:

```
complete("specdata", 1)
```

Out[7]:

	id	nobs
0	1	117

In [8]:

```
complete("specdata", [2, 4, 8, 10, 12])
```

Out[8]:

	id	nobs
0	2	1041
1	4	474
2	8	192
3	10	148
4	12	96

In [9]:

```
complete("specdata", "30:25")
```

Out[9]:

	id	nobs
0	25	463
1	26	586
2	27	338
3	28	475
4	29	711
5	30	932

Part 3

Write a function named corr that takes a directory of data files and a threshold for complete cases and calculates the correlation between sulfate and nitrate for monitor locations where the number of completely observed cases (on all variables) is greater than the threshold. The function should return a vector of correlations for the monitors that meet the threshold requirement. If no monitors meet the threshold requirement, then the function should return a numeric vector of length 0.

In [10]:

```
def corr(dirname, threshold):
    sulf = []
    nitr = []
    for i in range(1, 332+1):
        if i < 10: i_str = '00' + str(i)
        if i >= 10 and int(i) < 100: i_str = '0' + str(i)
        if i >= 100: i_str = str(i)
        files = glob.glob(f'{dirname}/{i_str}.csv')
        dataset = pd.DataFrame()
        for file in files:
            data = pd.read_csv(file).dropna()
            amount = len(data.index)
            if amount >= threshold:
                sulf += data['sulfate'].tolist()
                nitr += data['nitrate'].tolist()
            else: continue
    if len(sulf) == 0 or len(nitr) == 0: return 0
    else: return np.corrcoef(np.array(sulf), np.array(nitr))[0,1]
```

In [11]:

```
corr("specdata", 150)
```

Out[11]:

```
0.06069887784423783
```

In [12]:

```
corr("specdata", 400)
```

Out[12]:

```
0.056808397067008534
```

In [13]:

```
corr("specdata", 5000)
```

Out[13]:

```
0
```