Author: Oluwaseun Alexander Dada
Email: sogalexis@gmail.com
Linkedin: https://www.linkedin.com/in/oadada/

This solution loads data (Chromosomes, Position, MAF) from two different files and calculates the Pearson correlation coefficient over overlapping MAF entries.

In [1]:

```
# importing the needed libraries
import pandas as pd
from scipy.stats import pearsonr
```

In [2]:

```
# relevant datasets are loaded
data1 = pd.read_csv("Dataset1.txt",sep='\t')
data2 = pd.read_csv("Dataset2.txt",sep='\t')
```

In [3]:

```
# here is a snapshot of the dataset1
data1
```

Out[3]:

	Chromosome	Position	MAF
0	2	10587	0.025993
1	2	10596	0.026725
2	2	11274	0.030329
3	2	11276	0.039596
4	2	11320	0.162068
5	2	11336	0.037501
6	2	11357	0.037593
7	2	11486	0.037730
8	2	11607	0.037881
9	2	11834	0.037883
10	2	11842	0.044395
11	2	11985	0.070515
12	2	12154	0.039874
13	2	12220	0.027274
14	2	12240	0.027447
15	2	12320	0.036964
16	2	12371	0.034116
17	2	12444	0.037997
18	2	12618	0.037937
19	2	12697	0.027315
20	2	12994	0.048656
21	2	13133	0.033818
22	2	13241	0.120676
23	2	13651	0.037970
24	2	13750	0.038022
25	2	14485	0.037472
26	2	14983	0.034356
27	2	15491	0.038007
28	2	15562	0.037920
29	2	15672	0.027624
...	...	...	...
74	2	31791	0.086415
75	2	32003	0.039656
76	2	32005	0.065995

	Chromosome	Position	MAF
77	2	33012	0.034814
78	2	33159	0.027463
79	2	34049	0.061934
80	2	34503	0.033926
81	2	34831	0.027454
82	2	34930	0.033954
83	2	35015	0.027441
84	2	36326	0.013819
85	2	36787	0.109794
86	2	37100	0.033941
87	2	37136	0.027538
88	2	37394	0.033934
89	2	38101	0.033997
90	2	38219	0.110892
91	2	38733	0.109881
92	2	38938	0.076891
93	2	39340	0.109876
94	2	39356	0.069830
95	2	39555	0.121893
96	2	40569	0.110845
97	2	41366	0.120891
98	2	41404	0.109308
99	2	41720	0.027346
100	2	43092	0.037315
101	2	43210	0.033950
102	2	43362	0.033674
103	2	46411	0.021954

104 rows × 3 columns

In [4]:

```
# a snapshot of the dataset2
data2
```

Out[4]:

	Chromosome	Position	MAF
0	2	10514	0.011981
1	2	10515	0.020767
2	2	10554	0.154553
3	2	10560	0.154353
4	2	10566	0.154553
5	2	10571	0.011382
6	2	10573	0.117812
7	2	10574	0.060903
8	2	10587	0.132788
9	2	10595	0.053514
10	2	10596	0.141174
11	2	10597	0.011981
12	2	10649	0.057907
13	2	10662	0.019768
14	2	10682	0.057907
15	2	10705	0.057907
16	2	10751	0.019968
17	2	10797	0.040336
18	2	11247	0.157149
19	2	11274	0.098443
20	2	11276	0.161142
21	2	11288	0.012181
22	2	11320	0.163139
23	2	11336	0.156150
24	2	11343	0.084864
25	2	11357	0.155950
26	2	11361	0.016174
27	2	11392	0.075280
28	2	11406	0.010583
29	2	11486	0.158546
...	...	...	...
284	2	69022	0.065895
285	2	69431	0.189696
286	2	70074	0.238618

	Chromosome	Position	MAF
287	2	70187	0.028554
288	2	70190	0.032748
289	2	70325	0.083666
290	2	70436	0.076278
291	2	71018	0.071086
292	2	71045	0.107228
293	2	71196	0.079273
294	2	71631	0.012979
295	2	71695	0.056110
296	2	72184	0.232628
297	2	72221	0.025759
298	2	72475	0.030950
299	2	72515	0.073882
300	2	72730	0.010783
301	2	73297	0.012181
302	2	73635	0.012380
303	2	74387	0.112021
304	2	74483	0.065296
305	2	74588	0.074281
306	2	75383	0.069689
307	2	75741	0.034944
308	2	75947	0.023562
309	2	76379	0.363419
310	2	76417	0.412340
311	2	76420	0.391174
312	2	76422	0.341054
313	2	76485	0.165935

314 rows × 3 columns

Task A

Write a stand-alone program that can efficiently find all overlapping (identical) positions from these two files and outputs information for those overlapping (identical) positions into a tabulated output file that has 4 columns: Chromosome, Position, MAF1 (from Dataset1), MAF2 (from Dataset2).

Program should quit and give error if number of columns is wrong in either of the files.

In [5]:

```
# validation

# checking to ensure that the number of columns in the two datasets are equal
if (len(data1.columns) != len(data2.columns)):
    raise ValueError("Error: The number of columns in the two datasets must be equal!")

# checking to ensure that there are null entries in dataset1
if(data1.isnull().any().any()):
    raise ValueError("Error: Null not allowed in DataSet1", data1[data1.isnull().T.any().T]

# checking to ensure that there are null entries in dataset2
if(data2.isnull().any().any()):
    raise ValueError("Error: Null not allowed in DataSet2", data2[data2.isnull().T.any().T]
```

In [6]:

```
# overlapping (identical) positions in dataset1 and dataset2 are retrieved
# and stored in a new dataframe (data_merged)

chr = []
pos = []
maf_1 = []
maf_2 = []

for index1, row1 in data1.iterrows():
    for index2, row2 in data2.iterrows():
        isChromosomeEqual = row1['Chromosome']== row2['Chromosome']
        isPositionEqual = row1['Position']== row2['Position']

        if(isChromosomeEqual & isPositionEqual):
            chr.append(row2['Chromosome'])
            pos.append(row2['Position'])
            maf_1.append(row1['MAF'])
            maf_2.append(row2['MAF'])

data_merged = pd.DataFrame(list(zip(chr, pos, maf_1, maf_2)), columns=['Chromosome','Positi
```

In [7]:

```
# put data in the right format
data_merged['Chromosome'] = data_merged['Chromosome'].astype(int)
data_merged['Position'] = data_merged['Position'].astype(int)
data_merged['MAF_1'] = data_merged['MAF_1'].round(7)
data_merged['MAF_2'] = data_merged['MAF_2'].round(7)
```

In [8]:

```
# here is a snapshop of the tabulated output containing the 4 columns:
# Chromosome, Position, MAF1 (from Dataset1), MAF2 (from Dataset2)
data_merged
```

Out[8]:

	Chromosome	Position	MAF_1	MAF_2
0	2	10587	0.025993	0.132788
1	2	10596	0.026725	0.141174
2	2	11274	0.030329	0.098443
3	2	11276	0.039596	0.161142
4	2	11320	0.162068	0.163139
5	2	11336	0.037501	0.156150
6	2	11357	0.037593	0.155950
7	2	11486	0.037730	0.158546
8	2	11607	0.037881	0.164137
9	2	11834	0.037883	0.159345
10	2	11842	0.044395	0.205871
11	2	11985	0.070515	0.224840
12	2	12154	0.039874	0.161741
13	2	12220	0.027273	0.017772
14	2	12240	0.027447	0.017772
15	2	12371	0.034116	0.064097
16	2	12444	0.037997	0.161342
17	2	12618	0.037937	0.149960
18	2	12697	0.027315	0.042732
19	2	12697	0.027315	0.022764
20	2	12994	0.048656	0.171126
21	2	13133	0.033818	0.060503
22	2	13241	0.120676	0.030152
23	2	13651	0.037970	0.156350
24	2	13750	0.038022	0.164337
25	2	14485	0.037472	0.137181
26	2	14983	0.034356	0.134585
27	2	15491	0.038007	0.161142
28	2	15562	0.037920	0.154353
29	2	15672	0.027624	0.036741
...	...	...	...	...
64	2	30740	0.027325	0.016973
65	2	30762	0.702882	0.189896

	Chromosome	Position	MAF_1	MAF_2
66	2	30940	0.110863	0.219848
67	2	31221	0.027424	0.029952
68	2	31318	0.083234	0.189896
69	2	31324	0.083331	0.189497
70	2	31791	0.086415	0.228035
71	2	32005	0.065995	0.163139
72	2	33012	0.034814	0.090056
73	2	34049	0.061934	0.147165
74	2	34503	0.033926	0.073682
75	2	34831	0.027454	0.036342
76	2	35015	0.027441	0.030152
77	2	36787	0.109794	0.220048
78	2	37100	0.033941	0.081470
79	2	37136	0.027538	0.033347
80	2	37394	0.033934	0.079273
81	2	38101	0.033997	0.092053
82	2	38219	0.110892	0.220647
83	2	38938	0.076891	0.130591
84	2	39340	0.109876	0.220647
85	2	39356	0.069830	0.076877
86	2	39555	0.121893	0.026757
87	2	40569	0.110845	0.236621
88	2	41366	0.120891	0.025160
89	2	41404	0.109308	0.188498
90	2	41720	0.027346	0.016773
91	2	43092	0.037315	0.149161
92	2	43210	0.033950	0.084066
93	2	43362	0.033674	0.078075

94 rows × 4 columns

In [9]:

```
# write merged dataset to file
data_merged.set_index('Position', inplace=True)
data_merged.to_csv("correlation_output.txt", sep='\t')
```

Task B

Write a program that takes the output of the Task A (Chromosome, Position, MAF1 and MAF2) and calculates Pearson correlation coefficient over all corresponding MAF-values (MAF1 vs. MAF2) and outputs this correlation coefficient.

In [10]:

```
# calculate Pearson correlation coefficient
# Pearson correlation coefficient measures the linear relationship between two datasets: ma

correlation = pearsonr(maf_1, maf_2)
correlation
```

Out[10]:

```
(0.3540206410007135, 0.0004643535443743488)
```

In [ ]:

```

```