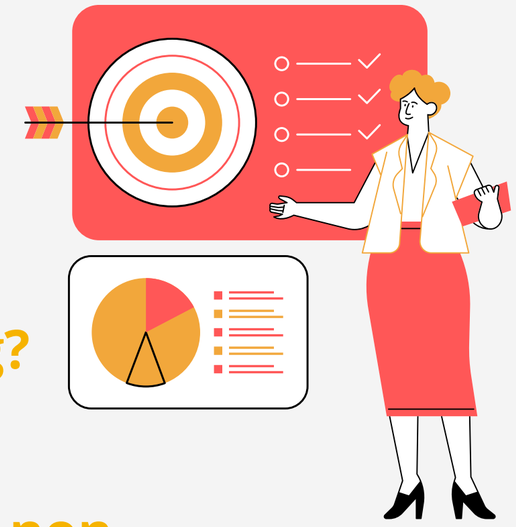
USING CLUSTERING, STATISTICAL MODELING, AND CLIMATE CHANGE PROJECTIONS TO ANALYZE RECENT AND FUTURE REGION-SPECIFIC COMPOUND OZONE AND TEMPERATURE BURDEN OVER EUROPE

Susanna Casarini, Gaetano Ferraiuolo e Marta Iacolino

Research questions



1. Can we identify region of similar ozone and temperature characteristics through clustering?
2. What are the differences between blended and non-blended temperature data?
3. How does our limited data affect our results compared to those obtained in the reference paper?
4. How can we visualize the different relationships between ozone and temperature in various regions?

Input data

Temperature (TX):
ECA data in blended and non-blended version

Ozone (O3):
Historic Airbase air quality data

Selected stations:
FR France (FR07004, FR26010)
GB UK (GB0033R, GB0567A)
IE Ireland (IE0001R)
NL Netherland (NL00807)

Time period:
from 2000 to 2012

IMPORTING DATA

Ozone:

```
datifr00_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr01_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr02_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr03_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr04_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr05_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr06_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr07_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr08_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr09_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr10_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr11_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
datifr12_26010 = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\pr1_41-main-dataset-FR\pr1_41-main-da
```

Temperature:

```
fr_clemont = pd.read_csv( r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\ECA_blend_tx\TX_STAID000750.txt' , deli
fr_dijon = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\ECA_blend_tx\TX_STAID000745.txt', delimite
ie_valentia = pd.read_csv(r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\ECA_blend_tx\TX_STAID000123.txt', delim

gb_penicuik =pd.read_csv( r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\ECA_blend_tx\TX_STAID001831.txt', delim
gb_stormont = pd.read_csv( r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\ECA_blend_tx\TX_STAID001821.txt', deli
nl_heino =  pd.read_csv( r'C:\Users\user\Desktop\modulo 2 Machine learning\project_1_41\ECA_blend_tx\TX_STAID000453.txt', delimi
```

CLEANING AND PROCESSING OF TX DATA

Temperature: *filtering of 2000-2012 period and mean computation*

```
# Filter years from 2000 to 2012 where "TX" is different from -9999
start_year = 2000
end_year = 2012
fr_clemont['    DATE'] = pd.to_datetime(fr_clemont['    DATE'], format='%Y%m%d')

fr_clemont_filtered = fr_clemont[
   (fr_clemont['    DATE'] >= f'{start_year}0101') & (fr_clemont['    DATE'] <= f'{end_year}1231') &
    (fr_clemont['   TX'] != -9999)
]

fr_clemont_mean = fr_clemont_filtered["   TX"].mean()


fr_dijon['    DATE'] = pd.to_datetime(fr_dijon['    DATE'], format='%Y%m%d')
fr_dijon_filtered = fr_dijon[
   (fr_dijon['    DATE'] >= f'{start_year}0101') & (fr_dijon['    DATE'] <= f'{end_year}1231') &
    (fr_dijon['   TX'] != -9999)
]

fr_dijon_mean= fr_dijon_filtered["   TX"].mean()
```

CLEANING AND PROCESSING OF TX DATA

Temperature: *definition of DataFrame with TX and DATE variables*

```
fr_clemont_filtered['    DATE'] = pd.to_datetime(fr_clemont_filtered['    DATE'])
fr_clemont_filtered['SoloGiorno'] = fr_clemont_filtered['    DATE'].dt.date
fr_clemont_filtered_TX = fr_clemont_filtered.set_index('SoloGiorno')
fr_clemont_filtered_TX['DATE'] = fr_clemont_filtered.index
colonne_da_tenere = ["   TX"]
fr_clemont_filtered_TX = fr_clemont_filtered_TX.filter(items=colonne_da_tenere)

fr_dijon_filtered['    DATE'] = pd.to_datetime(fr_dijon_filtered['    DATE'])
fr_dijon_filtered['SoloGiorno'] = fr_dijon_filtered['    DATE'].dt.date
fr_dijon_filtered_TX = fr_dijon_filtered.set_index('SoloGiorno')
fr_dijon_filtered_TX['DATE'] = fr_dijon_filtered_TX.index
colonne_da_tenere = ["   TX"]
fr_dijon_filtered_TX = fr_dijon_filtered_TX.filter(items=colonne_da_tenere)
```

BLENDED AND NON-BLENDED DATA COMPARISON

*Mean difference only in **Valencia station** (IE)*

	blended	nonblended
0	171.405138	171.405138
1	160.442198	160.442198
2	139.588252	136.787361
3	122.056230	122.056230
4	136.883044	136.883044
5	142.631501	142.631501

```
data = pd.DataFrame([(fr_clemont_mean_nonb, fr_clemont_mean),
                     (fr_dijon_mean_nonb,fr_dijon_mean),
                     (ie_valentia_mean_nonb,ie_valentia_mean),
                     (gb_penicuik_mean_nonb,gb_penicuik_mean),
                     (gb_stormont_mean_nonb,gb_stormont_mean),
                     (nl_heino_mean_nonb,nl_heino_mean)],
                    columns=['blended', 'nonblended'])

data
display(data)
```

TX AND O3 MEANS DATASET

```
data =  {
    "O3": pd.Series([meanNLtot, meanIEtot, meanfr_26010tot , meanfr_07004tot, meangb_firsttot, meangb_secondtot],
                    index=["NL", "IE", "FR26010", "FR07004", "GB_penicuik", "GB_stormont"]),

    "TX": pd.Series([nl_heino_mean, ie_valentia_mean, fr_clemont_mean, fr_dijon_mean, gb_penicuik_mean, gb_stormont_mean],
                    index=["NL", "IE", "FR26010", "FR07004", "GB_penicuik", "GB_stormont"])
}

dataset = pd.DataFrame(data)
```

	O3	TX
NL	44.221314	142.631501
IE	69.313498	136.787361
FR26010	60.768535	171.405138
FR07004	50.445344	160.442198
GB_penicuik	56.473585	122.056230
GB_stormont	40.625919	136.883044

CLUSTERING ANALYSIS

1

Clustering with TX and O3 data

2

Clustering with TX and MDA8O3 data

3

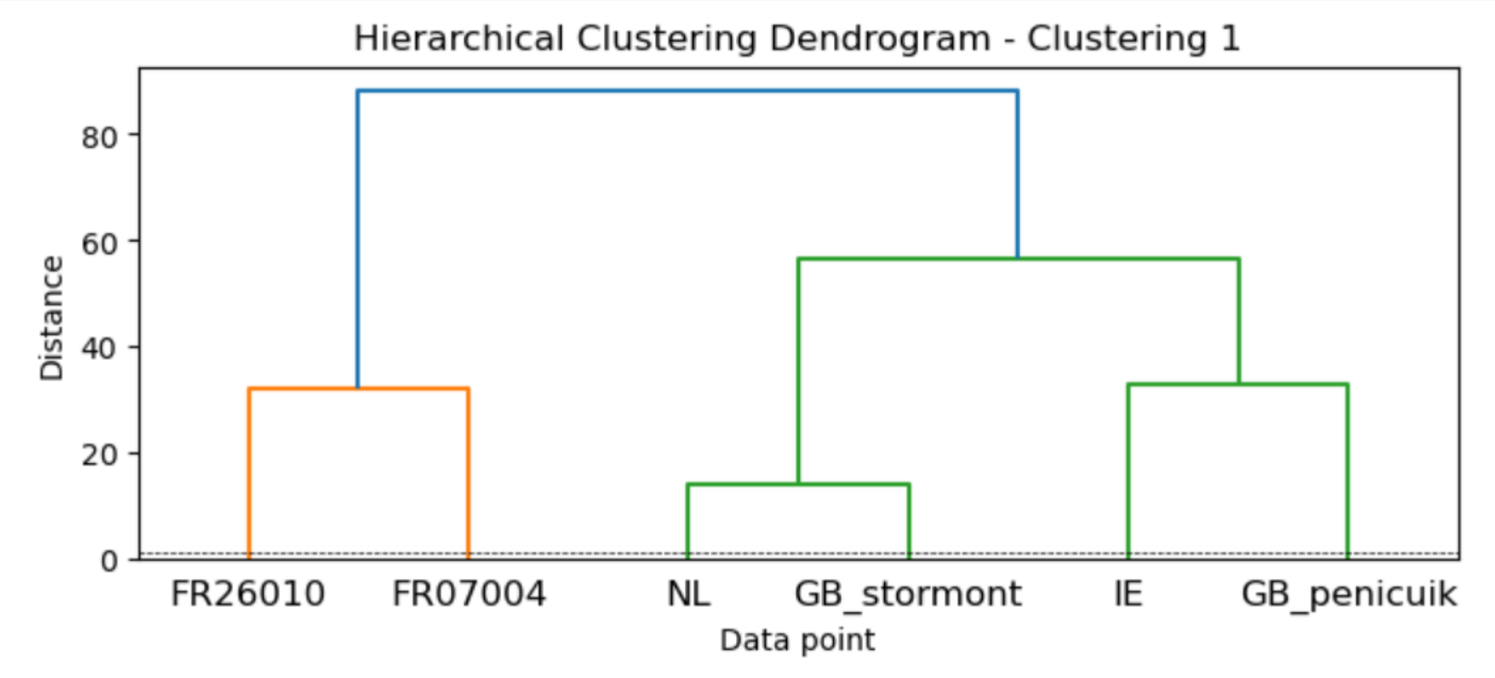
Clustering with TX and MDA8O3 data with missing data implementation

1

CLUSTERING 1 - TX AND 03

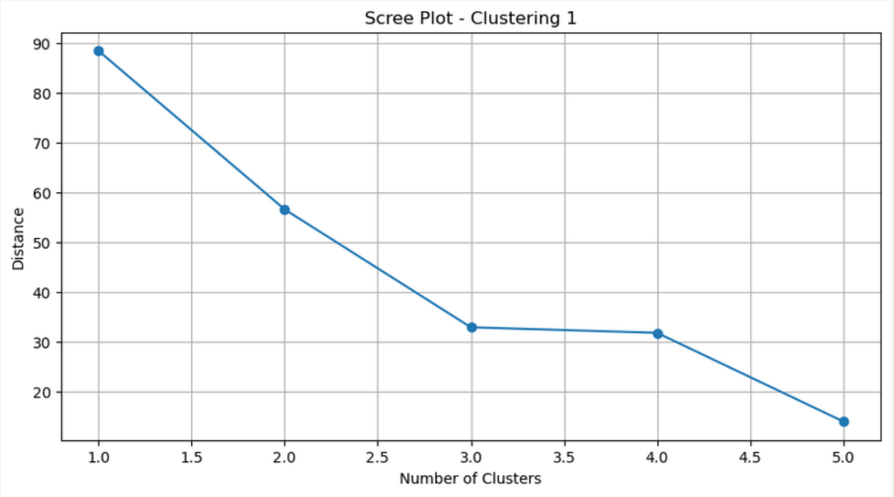
```
# 1. DISTANCE MATRIX
distances=pairwise_distances(dataset, metric= 'euclidean')
labels = ["NL", "IE", "FR26010", "FR07004", "GB_penicuik", "GB_stormont"]
distances_df = pd.DataFrame(distances,  columns=labels)

# 2. Z MATRIX
Z = linkage(distances, 'ward')
pd.DataFrame(Z, columns=['Document\Cluster 1', 'Document\Cluster 2',
                         'Distance', 'Cluster Size'], dtype="object")

# 3. DENDROGRAM:
plt.figure(figsize=(8, 3))
plt.title('Hierarchical Clustering Dendrogram - Clustering 1')
plt.xlabel('Data point')
plt.ylabel('Distance')
dendrogram(Z, labels=labels)
plt.axhline(y=1.0, c="k", ls="--", lw=0.5)
```

1 CLUSTERING 1 - TX AND 03



1 CLUSTERING 1 - TX AND 03

```
distances = Z[:, 2]
distances = distances[::-1]
num_clusters = np.arange(1, len(distances) + 1)

# scree plot
plt.figure(figsize=(10, 5))
plt.plot(num_clusters, distances, marker='o', linestyle='-')
plt.title('Scree Plot - Clustering 1')
plt.xlabel('Number of Clusters')
plt.ylabel('Distance')
plt.grid(True)
plt.show()
```



Scree plot: the “elbow” suggests to define three clusters:

1) France Clermont and France Dijon
2) Netherlands and Great Britain Stormont
3) Ireland and Great Britain Penicuik

2 CLUSTERING - 2 TAX AND MDA8O3

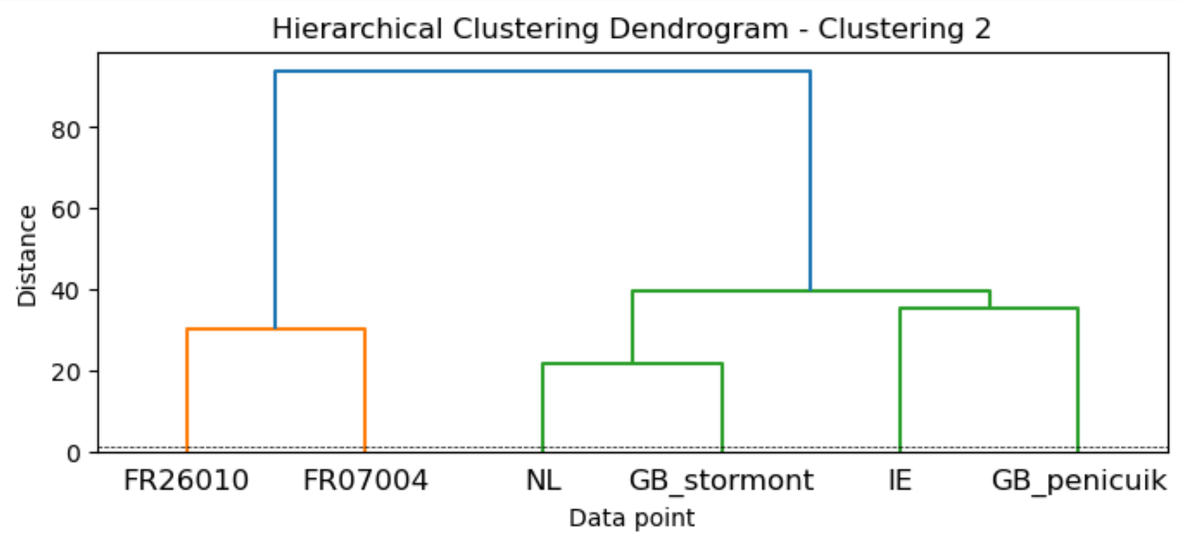
MDA8O3 computation

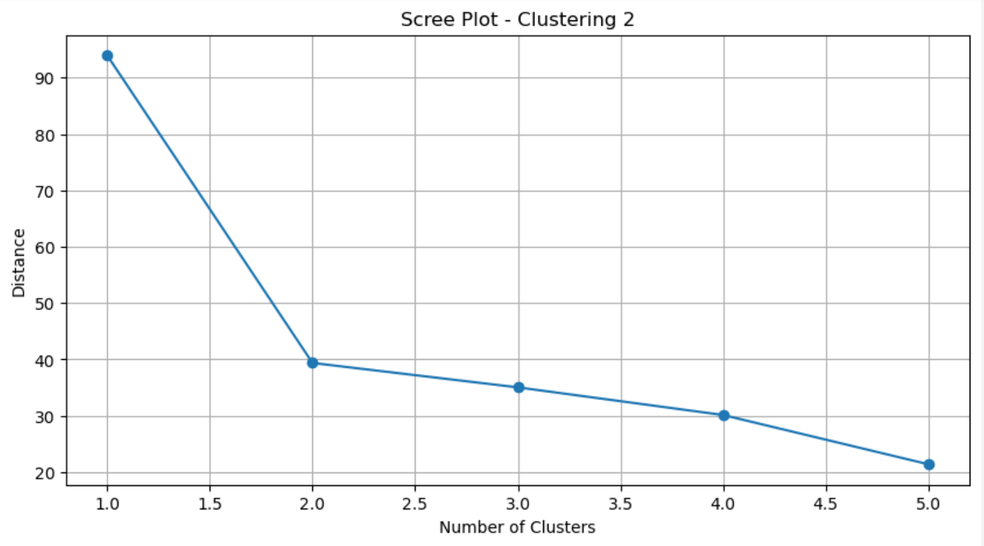
```
# cycle for calculating MDA8O3 for all the datasets of every station
def process_dataset(dataset):
    dataset['DatetimeBegin'] = pd.to_datetime(dataset['DatetimeBegin'])
    dataset['DatetimeEnd'] = pd.to_datetime(dataset['DatetimeEnd'])

    # Consider only misurations targeted as valid
    valid_data = dataset[dataset['Validity'] == 1]

    # calculating MOVING AVERAGES
    valid_data['MDA8O3'] = valid_data['Concentration'].rolling(window=8, min_periods=1).mean()

    # calculating daily max of MDA8O3
    daily_max_MDA8O3 = valid_data.groupby(valid_data['DatetimeBegin'].dt.date)['MDA8O3'].max()

    # Counting number of valid MOVING AVERAGES for every day
    valid_counts = valid_data.groupby(valid_data['DatetimeBegin'].dt.date)['MDA8O3'].count()

    # Filter original dataframe keeping only data of the valid days
    valid_counts.rename('NumH', inplace=True)
    valid_counts = pd.DataFrame(valid_counts)
    daily_max_MDA8O3 = pd.DataFrame(daily_max_MDA8O3)
    merged_data = pd.merge(daily_max_MDA8O3, valid_counts, on='DatetimeBegin', how='inner')
    return merged_data[merged_data['NumH'] >= 18]

datasets = [FR_26010, FR_07004, IE, NL, GB_first, GB_second ]
```

MDA8O3:
Daily maximum 8-hr running means calculated from Air Quality eReporting ozone pollution data, by only considering days with a minimum of 18 valid 8-hr running means available.

	MDA8O3	TX
NL	65.227884	142.631501
IE	80.847238	136.787361
FR26010	79.425263	171.405138
FR07004	72.416386	160.442198
GB_penicuik	68.353888	122.056230
GB_stormont	56.440559	136.883044

2 CLUSTERING - 2 TAX AND MDA8O3





Scree plot: the “elbow” suggests to define two clusters:

1) France Clermont and France Dijon

2) Netherlands, Great Britain Stormont, Ireland and Great Britain Penicuik

3 CLUSTERING 3 - TX AND MDA8O3 IMPLEMENTATION

```
#imputation of missing data: use the mean of the station
df_merged_5['FR_26010'].fillna(meanMDA_FR_26010, inplace = True)
df_merged_5['FR_07004'].fillna(meanMDA_FR_07004, inplace = True)
df_merged_5['IE'].fillna(meanMDA_IE, inplace = True)
df_merged_5['NL'].fillna(meanMDA_NL, inplace = True)
df_merged_5['GB_first'].fillna(meanMDA_GB_first, inplace = True)
df_merged_5['GB_second'].fillna(meanMDA_FR_26010, inplace = True)

# COMPLETE STANDARDIZED  DATASET :
MDA8O3_ST = df_merged_5.drop('DATE', axis=1)

scaler = StandardScaler() #from sklearn
MDA8O3_ST = pd.DataFrame(scaler.fit_transform(MDA8O3_ST), columns=MDA8O3_ST.columns)
MDA8O3_ST.index = ["MDA8O3"]*len(MDA8O3_ST["GB_second"])
```

For each variable:

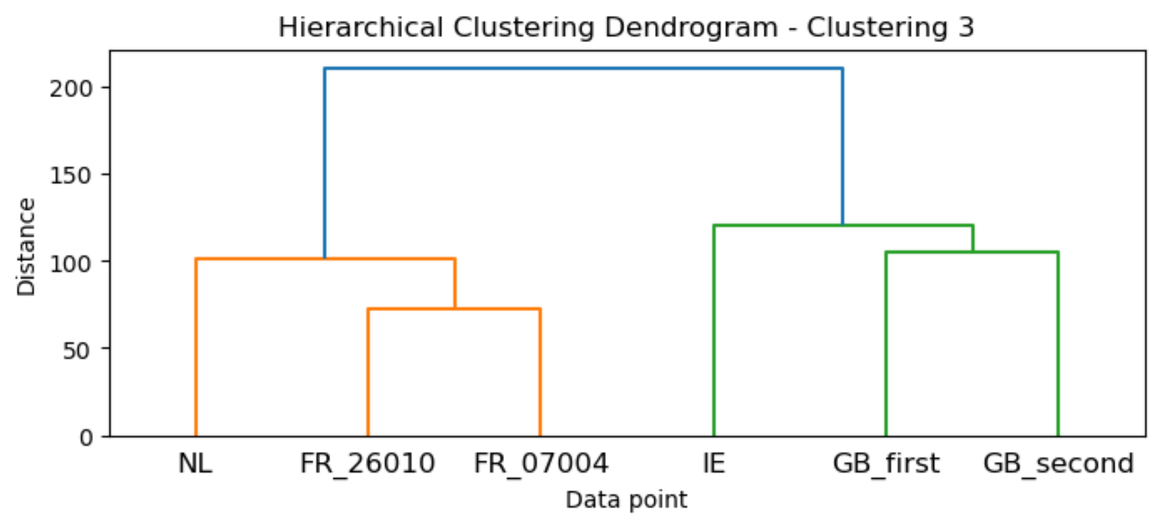
1. DataFrame with days in row and stations in column
2. Missing data imputation
3. Column standardization

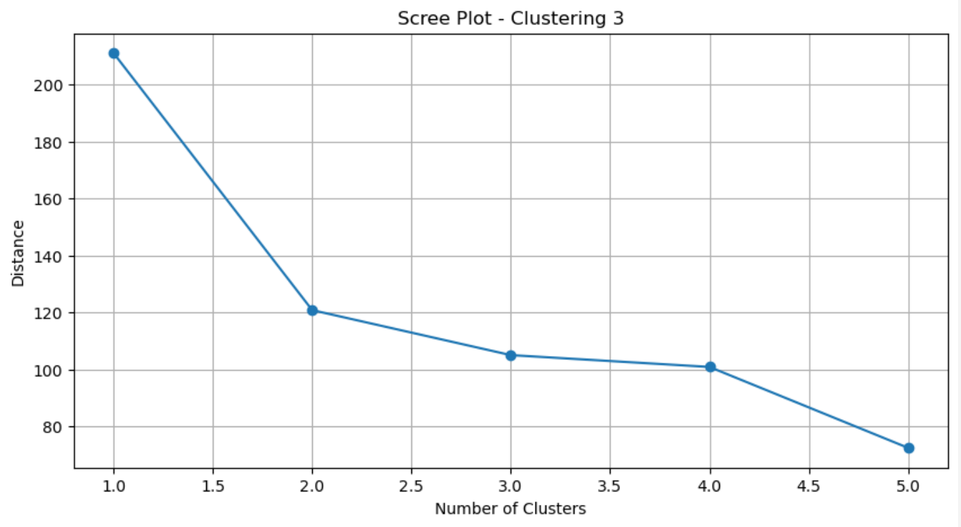
Clustering:

1. Concatenation of the two DataFrames
2. Clustering

```
# Union of MDA8O3 and TX
DATI_temporali = pd.concat([MDA8O3_ST, TX_ST], axis=0, ignore_index=False)
DATI_temporali_T = DATI_temporali.T

#CLUSTERING 3:
labels = ["FR_26010", "FR_07004", "IE", "NL", "GB_first", "GB_second"]

distances=pairwise_distances(DATI_temporali_T, metric= 'euclidean')
distances_df = pd.DataFrame(distances,  columns=labels)

Z = linkage(distances_df, method='ward')
```

3 CLUSTERING 3 - TX AND MDA8O3 IMPLEMENTATION





Scree plot: the “elbow” suggests to define two clusters:

1) France Clermont, France Dijon and Netherlands
2) Great Britain Stormont, Ireland and Great Britain Penicuik

SCATTER PLOT

DataFrames merging: *TX and MDA8O3 datasets merged for each station*

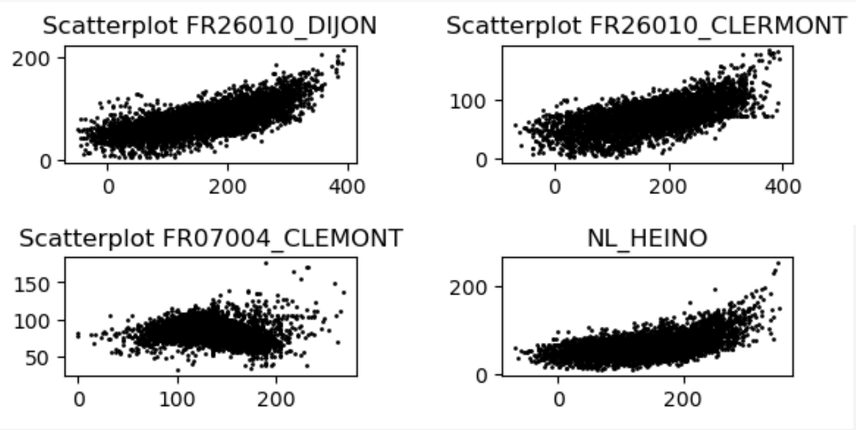
```
FR26010_DIJON = pd.merge(MDA8O3_FR_26010, fr_dijon_filtered_TX,
                         left_index = True, right_index = True, how = 'right')
FR26010_DIJON['MDA8O3'].fillna(meanMDA_FR_26010, inplace = True)
FR26010_DIJON = FR26010_DIJON.drop('NumH', axis=1)

#CLEMONT - FR07004:
FR07004_CLEMONT = pd.merge(MDA8O3_FR_07004 , fr_clemont_filtered_TX, left_index = True, right_index = True, how = 'right')
FR07004_CLEMONT['MDA8O3'].fillna(meanMDA_FR_07004, inplace = True)
FR07004_CLEMONT = FR07004_CLEMONT.drop('NumH', axis=1)
```

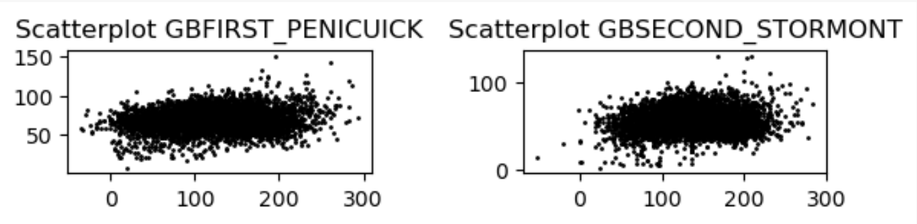
	MDA8O3	MDA8O3	MDA8O3	MDA8O3	MDA8O3	MDA8O3	MDA8O3	MDA8O3	MDA8O3
FR_26010	-1.068832e+00	-1.213742e+00	-1.420130e+00	-1.007355e+00	-1.525519e+00	-1.947076e+00	0.000000e+00	-1.494781e+00	-1.354261e+00
FR_07004	-1.661076e+00	-1.392942e+00	-2.263115e+00	-2.101223e+00	-1.954507e+00	5.751589e-16	5.751589e-16	-1.119747e+00	-1.529539e+00
IE	1.087184e-15	1.087184e-15	1.087184e-15	1.087184e-15	1.087184e-15	1.087184e-15	1.087184e-15	1.087184e-15	1.087184e-15
NL	-1.144358e+00	-1.428344e+00	-1.416294e+00	-5.609458e-01	-4.832817e-01	-1.412503e+00	-1.081293e+00	-7.185741e-01	-1.476290e+00
GB_first	-1.878879e-01	-1.878879e-01	-6.161373e-02	-6.161373e-02	-2.553540e-02	-1.157312e-01	-3.502403e-01	-3.141620e-01	-3.502403e-01
GB_second	-6.612628e-01	1.891293e-01	7.673959e-01	-3.040981e-01	6.313332e-01	9.034587e-01	5.973175e-01	3.081842e-01	-4.898048e-02

SCATTER PLOT

Scatter plot: *scatter plots for each station*

```
plt.subplot(3, 2, 1)
plt.plot(FR26010_DIJON['   TX'], FR26010_DIJON['MDA8O3'], marker='.', linestyle='None', color='black', markersize=2)
plt.title('Scatterplot FR26010_DIJON')

plt.subplot(3, 2, 2)
plt.plot(FR07004_CLEMONT['   TX'], FR07004_CLEMONT['MDA8O3'], marker='.', linestyle='None', color='black', markersize=2)
plt.title('Scatterplot FR26010_CLERMONT')

plt.subplot(3, 2, 3)
plt.plot(IE_VALENTIA['   TX'], IE_VALENTIA['MDA8O3'], marker='.', linestyle='None', color='black', markersize=2)
plt.title('Scatterplot FR07004_CLEMONT')
```





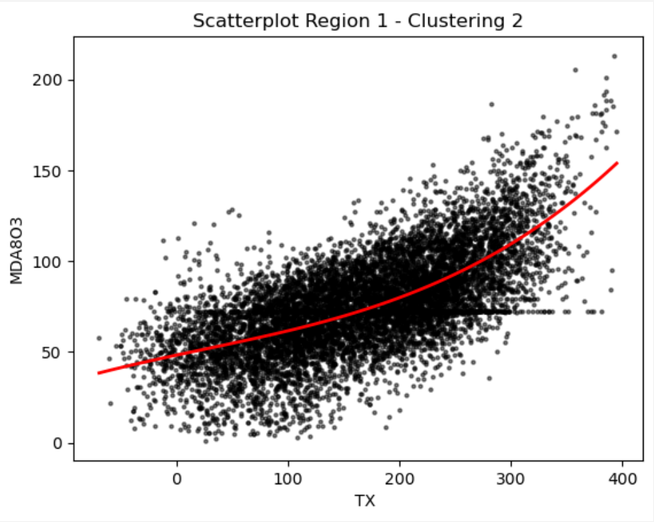
CLUSTERING 2

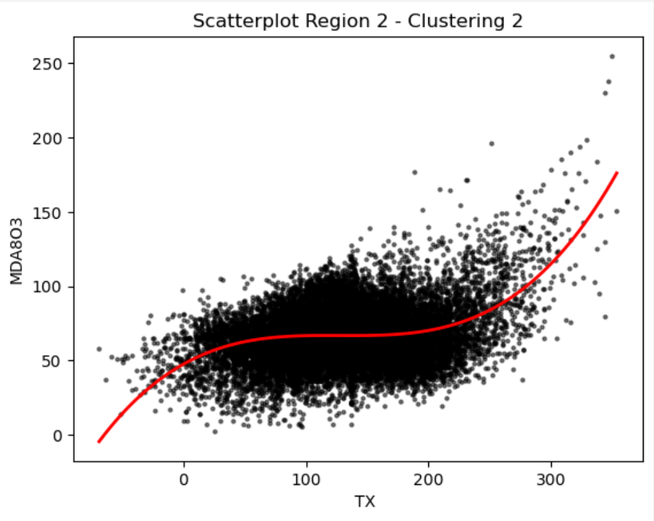
SCATTER PLOT

Scatter plots in clustering 2:

Scatter plot and polynomial computation

```
CLUSTER_1_second.plot(kind="scatter", x="   TX", y="MDA8O3" , c = "black",
                      marker = '.', alpha=0.5)

# Calculation of coefficients of polynomial of 2° grade
coefficients = np.polyfit(CLUSTER_1_second['   TX'], CLUSTER_1_second['MDA8O3'], 3)

# Creation of the polynomial using the coefficients
polynomial = np.poly1d(coefficients)

# Create a set of x values for tracing the polynomial in range min-max (100 numbers)
x_values = np.linspace(min(CLUSTER_1_second['   TX']),
                       max(CLUSTER_1_second['   TX']), 100)

# Tracing the polynomial
plt.plot(x_values, polynomial(x_values), color='red', linewidth=2)

# Adding a title to the graphic
plt.title('Scatterplot Region 1 - Clustering 2')
```

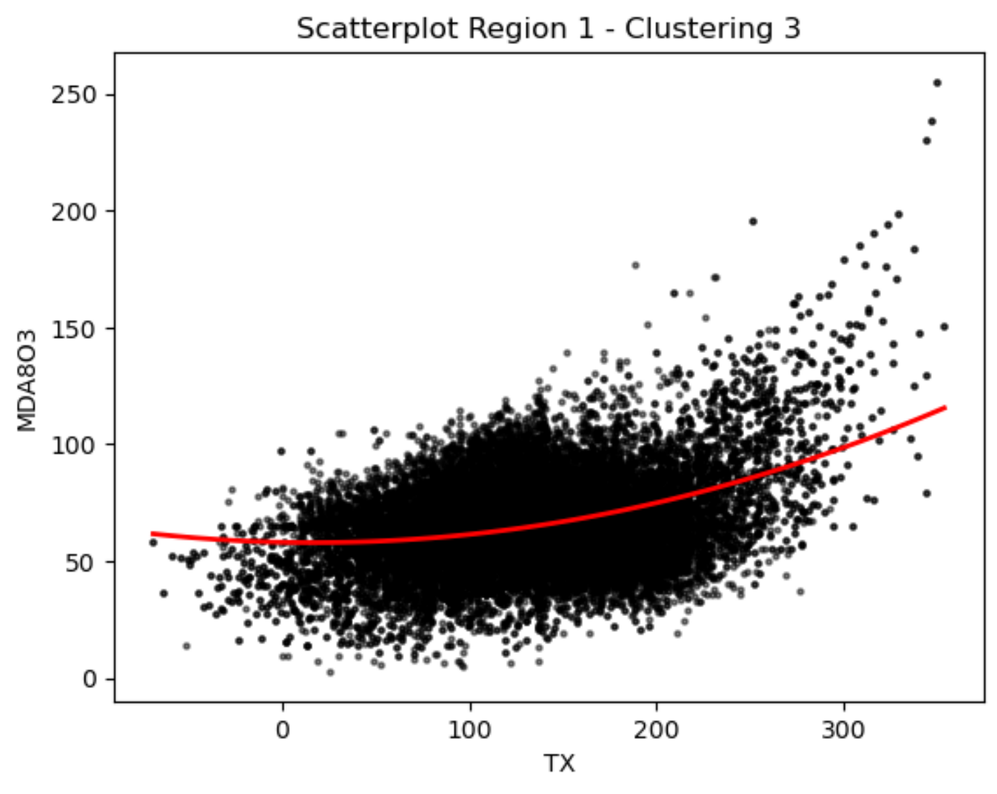


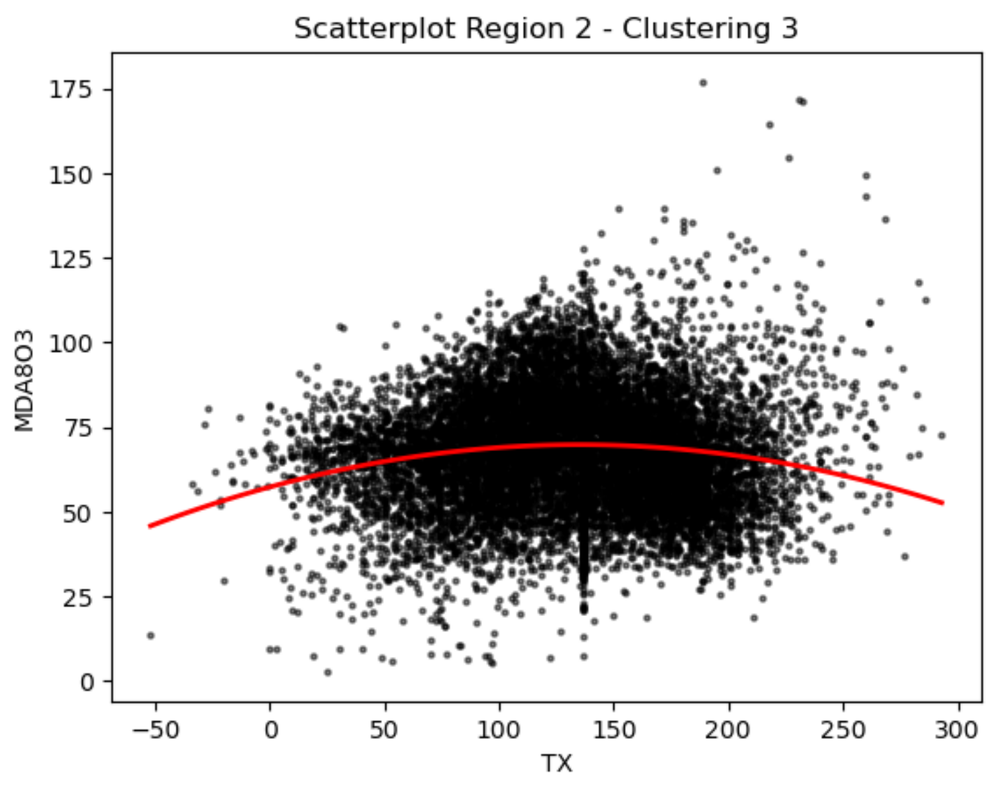


CLUSTERING 3

SCATTER PLOT

Scatter plots in clustering 2: *scatter plot and polynomial computation*



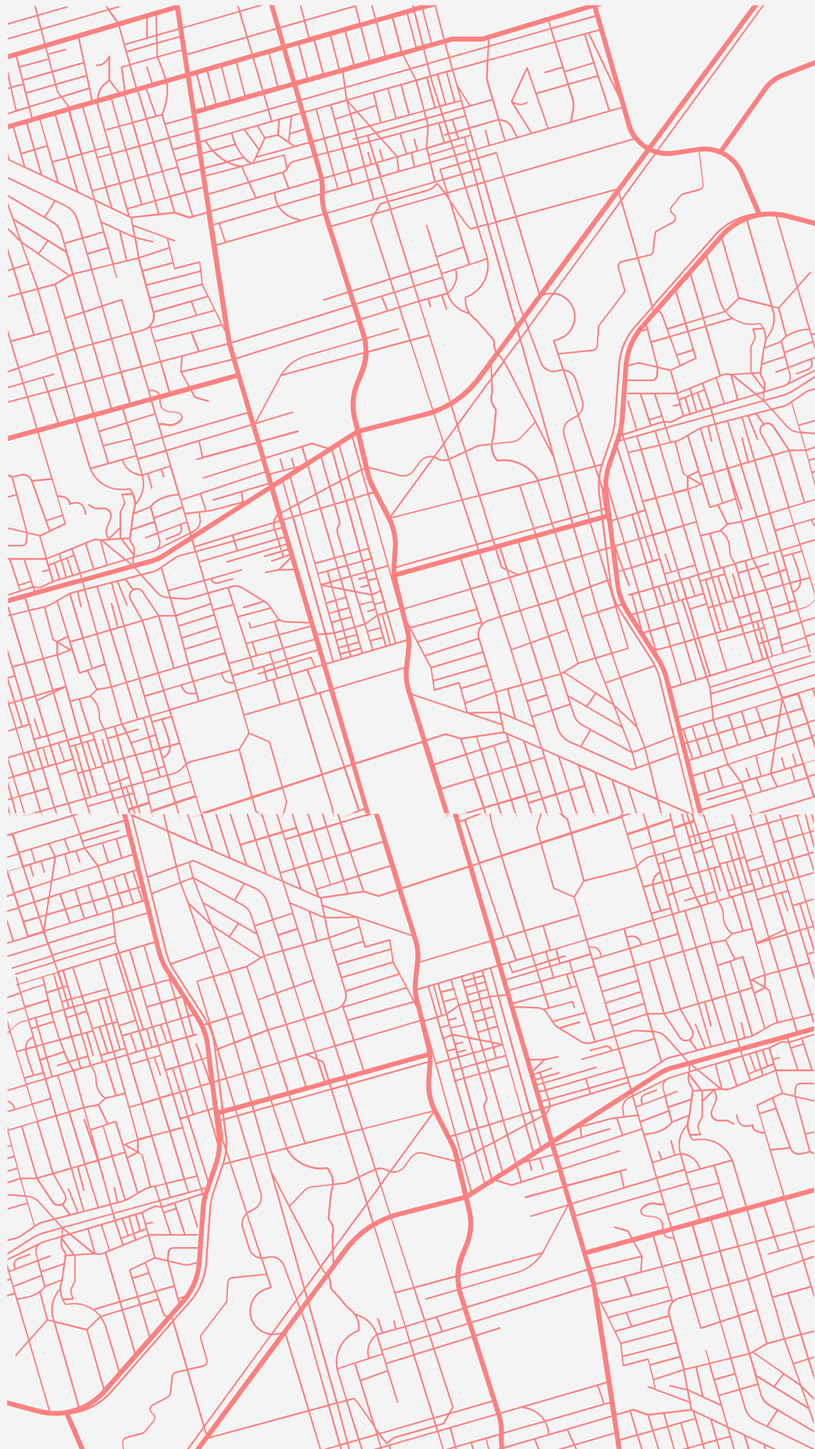


MAP OF EUROPE

Stations map: *regions visualization using different colours*

```
#for the 1° clustering:
coordinates = {
    'Stations': ["FR_26010", "FR_07004", "IE", "NL",
                 "GB_first", "GB_second"],
    'Longitude': [5.00, 3.11, -10.24, 6.40, -3.21, -5.93],
    'Latitude': [47.35, 45.80, 51.94, 52.39, 55.86, 54.60],
    'Cluster': [2,2,3,1,3,1], # this defines the clusters
    'Size': [20, 20, 20, 20, 20, 20]
}
coordinates = pd.DataFrame(coordinates)
```

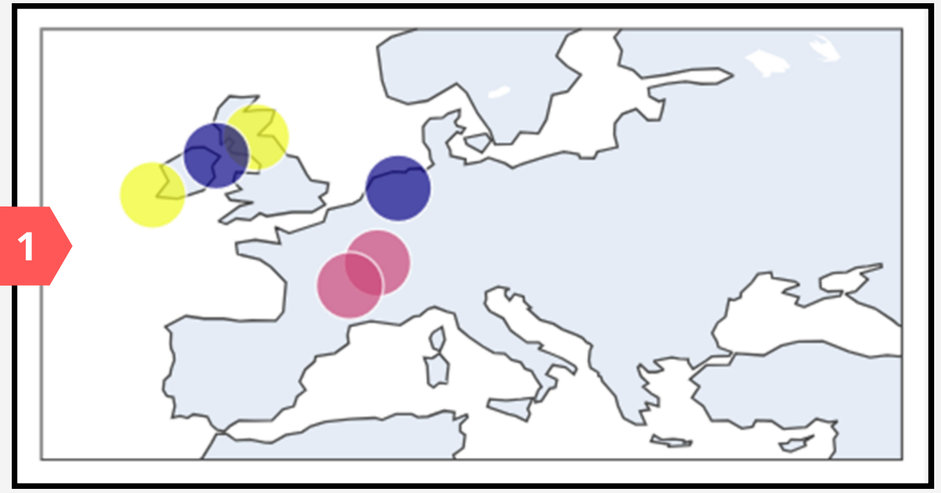
```
fig = px.scatter_geo(coordinates,
                     lat='Latitude',
                     lon='Longitude',
                     color='Cluster',
                     hover_name='Stations',
                     size= "Size",
                     projection='equirectangular',
                     title='Cluster Analysis in Europe - O3 Data')
fig.update_layout(coloraxis_colorbar=dict(title='Cluster', tickvals=[1, 2, 3],
                                ticktext=['Cluster 1', 'Cluster 2', 'Cluster 3']))
fig.show()
```

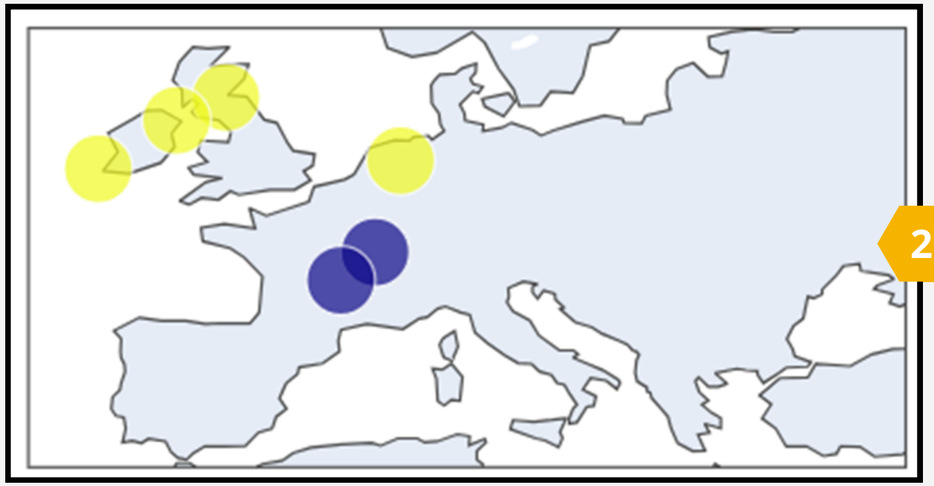


MAP OF EUROPE

CLUSTERING 1

1



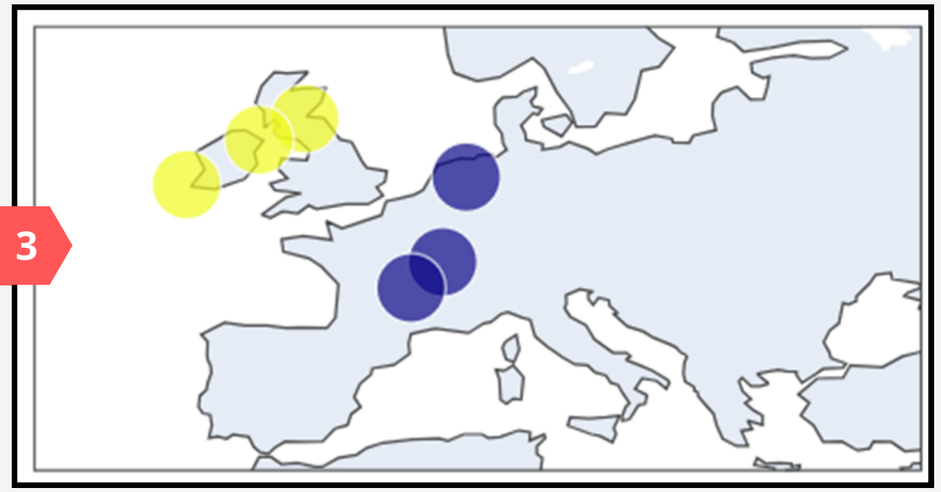


2

CLUSTERING 2

CLUSTERING 3

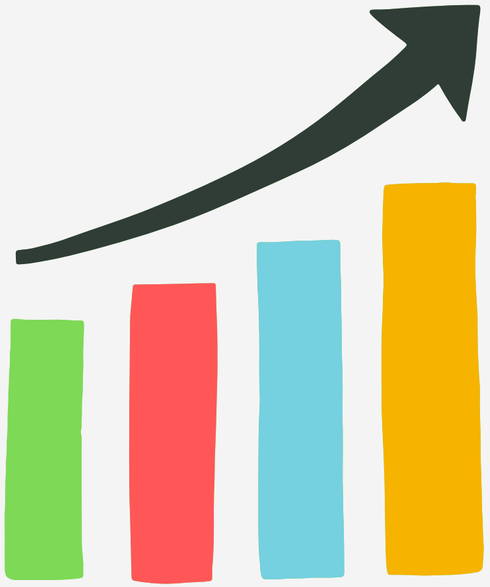
3



Summary statistics

Statistics for each region: *altitude, MDA8O3, TX*

```
Stats_clustering = {
    "Mean Altitude": pd.Series([336, 51.75], index = ["Region1", "Region2"]),
    "MeanMDA8O3": pd.Series([mean_MDA803_cluster1, mean_MDA803_cluster2],
                            index = ["Region1", "Region2"]),
    "MedianMDA8O3": pd.Series([percentile_50_MDA803_cluster1,
                 percentile_50_MDA803_cluster2],index = ["Region1", "Region2"]),
    "25thMDA8O3": pd.Series([percentile_25_MDA803_cluster1,
                  percentile_25_MDA803_cluster2], index = ["Region1", "Region2"]),
    "75thMDA8O3": pd.Series([percentile_75_MDA803_cluster1,
                percentile_75_MDA803_cluster2], index = ["Region1", "Region2"]),
    "MeanTX": pd.Series([mean_TX_cluster1, mean_TX_cluster2],
                        index = ["Region1", "Region2"]),
    "MedianTX": pd.Series([percentile_50_TX_cluster1, percentile_50_TX_cluster2],
                          index = ["Region1", "Region2"]),
    "25thTX": pd.Series([percentile_25_TX_cluster1, percentile_25_TX_cluster2],
                        index = ["Region1", "Region2"]),
    "75thTX": pd.Series([percentile_75_TX_cluster1, percentile_75_TX_cluster2],
                        index = ["Region1", "Region2"])
    }
Stats_clustering = pd.DataFrame(Stats_clustering)
display(Markdown("Summary Statistics - Clustering 2"))
display(Stats_clustering)
```



Summary statistics

Clustering 2

	Mean Altitude	MeanMDA8O3	MedianMDA8O3	25thMDA8O3	75thMDA8O3	MeanTX	MedianTX	25thTX	75thTX
Region1	336.00	75.920825	73.000000	59.000000	91.125	165.923668	169.000000	104.0	230.0
Region2	51.75	67.464327	66.995125	54.621812	79.150	134.995144	136.787361	98.0	173.0

Clustering 3

	Mean Altitude	MeanMDA8O3	MedianMDA8O3	25thMDA8O3	75thMDA8O3	MeanTX	MedianTX	25thTX	75thTX
Region1	226.33	66.998479	66.00	53.330125	78.907	136.585786	136.787361	96.0	230.0
Region2	66.67	68.262829	68.25	56.750000	79.500	132.268652	136.787361	101.0	167.0

Conclusions

We obtained clusters that represents regions with similar characteristics for ozone and temperature. Stations united by clustering are often geographically close

Choosing blended or non blended data for temperature does not affect the results, while different clustering approaches produce slightly different results

Having little data, we obtain fewer clusters compared to the paper, but those obtained are similar to those of the paper

From the scatterplots we note that in some regions the relationship between temperature and ozone does not seem evident, but an increase in ozone levels is noted for high temperature levels