DATA ANALYSIS OF AIR QUALITY IN GRONINGEN

Nishit Patel
n.t.patel@student.utwente.nl

STUDENT NO. – s2872129
COURSE – Acquisition & Exploration of Geo-Data
SPECIALIZATION – GeoInformatics
DATE – 01/02/2022

Contents

Figures

Scripts

Tables

1.Introduction

Background:

Air Pollution is one of the major issues that the world is facing right now. The harmful effects on human health caused due to air pollution have been documented in numerous amounts of studies and research. According to World Health Organization (WHO), Air Pollution causes about 7 million deaths each year (WHO, 2021). The main components contributing to air pollution and responsible for harmful effects on human health are CO, NO – NO_2, SO_2, and $PM_{2.5}$ – PM_{10} *(Particulate Matter)*. Studies showing direct correlation between these components and human health risks such as chest pain, inflammation, irritation of breathing passages, breathing difficulties for people with asthma and heart disease, and reduced lung function have been documented by U.S. Environmental Protection Agency (Nathanson, J. A., 2020). To help countries, avoid and mitigate these risks, WHO has published Global Air Quality Guidelines based on scientific evidence collected from all over the globe. Similarly, EU Ambient Air Quality Directives (AAQDs) also encourage and oblige EU member states to monitor the air quality and take action to assess and manage the air quality in a way that mitigates the above-mentioned health risks of Air Pollution.

Standards: The standards set for Air Quality levels both by World Health Organization (WHO) and European Environment Agency (EEA) are mentioned in the table below:

Components	***Averaging Period***	***Standards***	
		WHO	EEA
NO2	Annual	40 μg/m^3	40 μg/m^3
PM2.5	Annual	10 μg/m^3	25 μg/m^3
PM10	Annual	20 μg/m^3	40 μg/m^3
SO2	1 day	20 μg/m^3	125 μg/m^3
CO	Maximum daily 8-hour mean	10 mg/m^3	10 mg/m^3

Table 1: Air Quality Standards

With the above standards and background in mind, this report describes the analysis performed for two of the Air Quality components – $PM_{2.5}$ and NO_2 in the city of Groningen located in the north of the Netherlands. This report along with the performed analysis provides data and maps that the local government and the citizens can use to make informed decisions. In the same context, the results are provided by keeping the following questions in mind:

1. *What is the current and the mean quality of the air in the various districts of the city?*
2. *What is the air quality throughout the day in and around the green and recreational areas of the city?*
3. *What is/are the best routes for running in the city?*

2. Methodology

Data Identification: A time frame of January 2021 – November 2021 was considered for data analysis. Consequently, the data for this time frame was obtained via two major Dutch Air Quality monitoring networks. The first one called the 'Dutch Air Quality Monitoring Network *(Lucht-MeetNet)*' is a network of high-quality sensors set up by various governmental organizations in various provinces and major cities of the Netherlands. In the context of the city of Groningen and components $PM_{2.5}$ and NO_2, there are a total of 2 working sensors measuring the values of these components on hourly basis. Even though the data being high quality, it was not enough to tell the whole story for the entire city of Groningen. Thus, additionally, 10 sensors measuring $PM_{2.5}$ and 2 sensors measuring NO_2 were selected from the 'Measuring Together *(Samen-Meten)*' initiative set up by the National Institute for Public Health and the Environment (RIVM). The figure below shows the location of the network of sensors selected for this analysis.

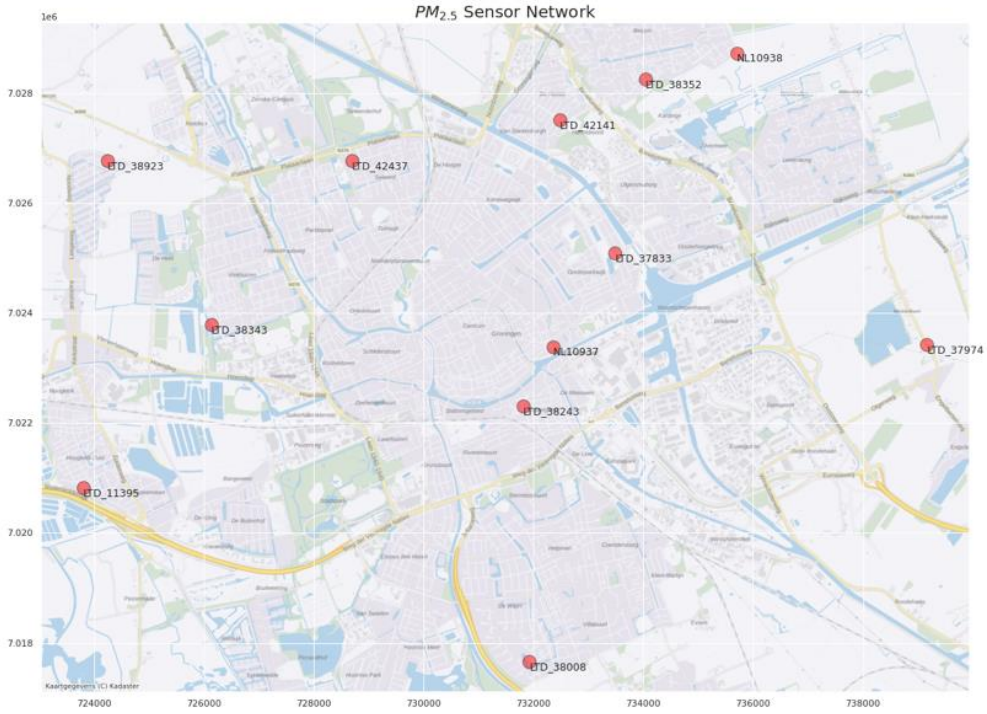


Figure 1: Selected Sensors for $PM_{2.5}$ Analysis

The script used to get the above plot:

Script 1: Plotting the selected sensors for Data Analysis on a Basemap

```
# run this script after running the scripts of from part 1 and part 2 of Data Analysis Section
print()

# basemap library import
import contextily as cx
```

```
# setting up coordinate reference system for the geopandas dataframe
gdf.crs = 'EPSG:4326'

# converting the coordinate reference system to that of the basemap used below
map_df = gdf.to_crs(epsg = 3857)

# styling and plotting the sensors
axg = map_df.plot(figsize=(20, 15), alpha=0.5, edgecolor='k', markersize=250,
            color='red')
map_df.apply(lambda x: axg.annotate(text=x['station'],
                        xy=x.geometry.centroid.coords[0],
                        ha='left', va='top', size='12.5'), axis=1)
axg.set_title(r'$PM_{2.5}$ Sensor Network', fontsize=20)

# adding a basemap to the plot
cx.add_basemap(axg, source=cx.providers.nlmaps.standaard,
           alpha=0.5)


# saving the plot as png
fig = axg.get_figure()
fig.savefig('PM2.5_Sensor_Network.png')
```

(Note - Please note that the script takes the geopandas dataframe as an input which will be created after running the scripts mentioned in the Annual Analysis Segment.)

Similarly,

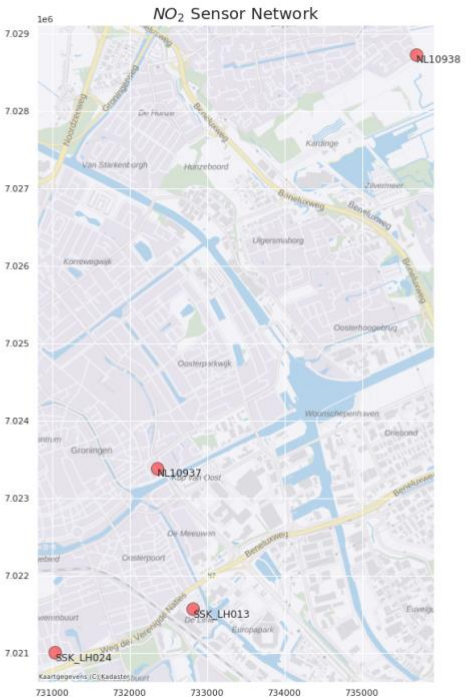


Figure 2: Selected Sensors for NO_2 Analysis

Data Analysis: The entire data analysis process was divided into three main parts – Annual Analysis, Selected Day Analysis and QGIS Processing.

Annual Analysis: The Annual Analysis was performed by keeping in mind the annual mean standards and questions described in the 'Introduction' Section. The Analysis was split into two main parts pertaining to the two selected sensor networks.

- ***Lucht-MeetNet Sensors:*** The data obtained from the *'Lucht-MeetNet'* network was in csv format. There were a total of 11 csv files which contained hourly measured values of all the measuring stations of the Netherlands. Thus, to filter out relevant data related to the study area of Groningen, a python script was developed that loops over all the input csv files and filters the data by given 'station-id'. Additionally, the python script also calculated the mean of the measured values for each day and stored it in a dataframe to facilitate further data operations and to plot a calendar heatmap of the time frame of the study. This python script is shown in the box below.

Script 2: Analysis and Management of the downloaded Data from 'Lucht-MeetNet' Network

```
# library imports
import csv
import pandas as pd
import glob
import matplotlib.pyplot as plt
import seaborn as sns

# empty list and dictionary to be filled later
collist = []
coordict_M = {}

# sensor list containing 'station-id' of interest
sensorlist = ['NL10937', 'NL10938']

for sensors in sensorlist:

    # empty mean dictionary to be used later
    meandict = {'mean_value': []}

    # identifying csv files from a specific path
    path = 'PM2.5_CSV/'        # point to the NO2 files to get the NO2 values
    all_files = sorted(glob.glob(path + '/*.csv'))

    month = ['January', 'February', 'March', 'April', 'May', 'June',
             'July', 'August', 'September', 'October', 'November']
    mo = 0

    # list used to transfer values to new dataframe
    transferlist = []

    # iterating over the identified csv files
    for file in all_files:
        f = open(file, encoding='latin-1')
        csv_reader = csv.DictReader(f, delimiter=';')

        # script to put in the elements of the csv file to an empty list
        emptylist = []
        for row in csv_reader:
            emptylist.append(row)
```

```
    y = emptylist
    station_id = sensors
    valuelist = []
    datetime = []

    # script to store the coordinates from the empty list into a dictionary
    for keys, valuess in emptylist[1].items():
        if keys == station_id:
            valuesplit = valuess.split(',')
            coordict_M.update({station_id:
                               [float(valuesplit[1][:-1]),
                                float(valuesplit[0][1:])]
                              }
                             )

    # script to get measured values and time of the station of interest
    for i in range(8, len(emptylist)):
        z = emptylist[i]
        for j in range(len(z)):
            datetime.append(z[''] + ' - ' + z['StationsCode'])
            if station_id in z:
                valuelist.append(z[str(station_id)])
                break

    # storing the above values in a dataframe
    dff = pd.DataFrame(valuelist, columns = ['value'])
    dff['Date_Time'] = datetime
    dff['value'] = pd.to_numeric(dff['value'], downcast='float')

    # calculating the mean of each day and storing it in 'meanlist'
    rowi = 0
    rowj = 23
    meanlist = []
    for index, rows in dff.iterrows():
        if rowj <= len(dff):
            dff_I = dff.loc[range(rowi, rowj)]
            meanlist.append(dff_I['value'].mean())
            rowi = rowj
            rowj += 24

    # storing the mean values in a new data frame
    # this was done to arrange the mean list...
    # based on the days (index of the dataframe)
    dff_HM = pd.DataFrame(meanlist, columns=['mean_value'])
    dff_HM.index += 1
    transferlist.append(dff_HM['mean_value'].values)

# transfering the values to a plot dictionary where...
# the values are categorized by months
plotdict = {}
for months in month:
    plotdict[months] = []

for itemss in transferlist:
    for itemsss in itemss:
        plotdict[month[mo]].append(itemsss)
    mo += 1

# converting the dictionary to a dataframe which will be used to plot the..
# values in form of a monthly calendar heatmap
monthdf = pd.DataFrame(dict([(k, pd.Series(v)) for k, v in plotdict.items()]))
```

```
monthdf.index += 1
print('station', '-', station_id)
print(monthdf.head())    # only printing first few values
print()

# extra lists which contains information about 'station-id' and annual mean
# these will be used later for interpolation
collist.append(station_id)
coordict_M[station_id].append(monthdf.mean().mean())

# the below script gives a monthly calendar heatmap
# this is shown in the 'Results' section of the report
sns.set(rc = {'figure.figsize':(12, 15)})
ax = sns.heatmap(monthdf, yticklabels = 2, cmap = 'RdYlGn_r')
ax.set_ylabel('Day')
ax.set_xlabel('Month')
ax.invert_yaxis()
ax.set_title(r'$PM_{2.5}$ Heatmap, Station - ' + station_id, fontsize = 15)
plt.show()
# fig = ax.get_figure()
# fig.savefig(r'$PM_{2.5}$ ' + station_id + 'heatmap.png')
```

- Thus, by the end of the above script all the csv files were filtered for relevant values using 'station-id'. Also, a dictionary was developed with contained both coordinates and annual mean values for every selected station. Further, a monthly calendar heat-map was also produced using the resulting dataframe to facilitate visual analysis *(included in 'Results' section).*

- ***Samen-Meten Sensors (Low-Cost):*** As discussed in the *'Data Identification'* Section, the data from the 'Measuring Together' initiative was also used for the analysis. This data was also obtained in 'csv' format. However, since the measuring sensors were 'low-cost', there were a lot of limitations related to the data *(limitations discussed in the 'Discussion' Section)*. However, unlike the data obtained as 'csv' from the *'Lucht-MeetNet'* network, here the 'csv' files were much simpler to deal with as there was a csv file per sensor, and it only contained two columns – 'time of measurement' and 'measurement'. Thus, again, a python script was developed to loop over all the csv files, store relevant information and calculate the annual mean per sensor.

Script 3: Analysis and Management of the downloaded Data from 'Samen-Meten' Network

```
# library imports
import csv
import pandas as pd
import glob
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# empty mean dictionary to be used later
meandict = {'mean_value':[]}

# identifying csv files from a specific path
path = 'MTPM25CSV/'        # point to the NO2 files to get the NO2 values
all_files = sorted(glob.glob(path + '/*.csv'))

# storing the name (id) of the sensor in a list
```

```
sensorlist = []

newtransferlist = []
for filename in all_files:
    namesplit = filename.split('/')
    sensorlist.append(namesplit[1][0:9])

print('selected sensors', '-', sensorlist)
print()

# iterating over the csv files
iterr = 0
for file in all_files:
    f = open(file, encoding='latin-1')
    csv_reader = csv.DictReader(f, delimiter=';')

    # taking all the values of a csv file and storing it to a list
    emptydict = {}
    dflist = []
    datelist = []
    for row in csv_reader:
        emptydict.update(row)
        for k, v in emptydict.items():
            strsplit = v.split(',')
            if strsplit[1] == '':
                strsplit[1] = 0
                dflist.append(float(strsplit[1]))
            else:
                dflist.append(float(strsplit[1]))
            datelist.append(strsplit[0])

    # appending the list with measured values to a new transfer list
    # this list will be converted into a dataframe
    newtransferlist.append(dflist)

# finding the list with maximum length amongst the other list
# this deals with the problem of difference in data availability for sensors
list_length = [len(dflist) for dflist in newtransferlist]
link_position = np.argmax(np.array(list_length))

# linking the value list with the sensor list and storing it to a dataframe
for lp in range(len(newtransferlist)):
    if lp == link_position:
        df_lowcost = pd.DataFrame({sensorlist[lp]: pd.Series(newtransferlist[lp])})
        df_lowcost.index += 1

for lp in range(len(newtransferlist)):
    if lp != link_position:
        df_lowcost[sensorlist[lp]] = pd.Series(newtransferlist[lp])

df_lowcost = df_lowcost.reindex(sorted(df_lowcost.columns), axis=1)
print(df_lowcost.head())              # printing first few values of the dataframe
print(df_lowcost.describe().transpose())  # statistics of the data frame


# creating a box plot which will be used later
boxplot = df_lowcost.boxplot(figsize = (20, 10))
boxplot.set(ylim = (-1, 20))
fig = boxplot.get_figure()
fig.savefig('PM2.5_Low-Cost_Box.png')
```

- The above script results into a dataframe containing information – (statistics) about all the selected sensors of 'Samen-Meten' Network. Now that there were two dataframes for two different sources, next step was to bring them together and calculate the annual mean values that would help make an interpolation map. To put the sensors' location on the map, coordinate values were also required. This was easy to get for the sensors of the '*Lucht-MeetNet'* network because the csv files contained the information about coordinates. Thus, this was done parallelly while developing ***Script 1***.

	count	mean	std	min	25%	50%	75%	max
LTD_11395	6356.0	4.170390	10.393928	0.0	2.0	2.0	4.0	424.0
LTD_37833	7420.0	4.758491	6.049814	0.0	1.0	3.0	7.0	196.0
LTD_37974	7828.0	9.482882	12.532919	0.0	4.0	6.0	11.0	204.0
LTD_38008	7902.0	6.111111	5.064580	0.0	3.0	5.0	8.0	80.0
LTD_38243	3640.0	4.231593	11.435444	0.0	1.0	3.0	5.0	543.0
LTD_38343	7909.0	2.570616	2.561776	0.0	1.0	2.0	3.0	32.0
LTD_38352	7801.0	8.443277	7.327549	0.0	4.0	7.0	11.0	263.0
LTD_38923	6275.0	7.330837	8.296498	0.0	3.0	5.0	9.0	260.0
LTD_42141	7232.0	5.140763	5.620639	0.0	2.0	3.0	6.0	63.0
LTD_42437	7209.0	4.098349	5.929587	0.0	2.0	3.0	5.0	187.0

Figure 3: Statistics ($PM_{2.5}$) of the selected sensors from the 'Samen-Meten' Network

- However, for the '*Samen-Meten*' sensor network, the csv files didn't contain any information about the coordinates of the sensor locations. This information was also not mentioned on the website. Thus, a script was used to get the coordinate values from API and add it to the dictionary of the '*Lucht-MeetNet*' Network created during ***Script 1***. The script is mentioned below:

Script 4: Getting coordinate values for the sensors of 'Samen-Meten' Network

```
# library imports
import urllib3
import json

http = urllib3.PoolManager()

# creating a custom url to filter out locations in a particular polygon area
url = "https://api-samenmeten.rivm.nl/v1.0/Locations?$filter=st_intersects(location, "
url += "geography'POLYGON((6.646072 53.245780, 6.677191 53.185053, 6.641361 53.159123, "
url += "6.561241 53.146227, 6.492015 53.157271, 6.412582 53.166608, 6.367357 53.200069, "
url += "6.344730 53.237277, 6.376067 53.281637, 6.567162 53.287902, 6.719266 53.273015, "
url += "6.777630 53.244263, 6.646072 53.245780))')"

request = http.request('GET', url)

data = json.loads(request.data.decode('utf-8'))

# iterating over the acquired data
for record in data['value']:
    # filtering only relevant sensors from all records and updating the coordinate dictionary
    if record['name'][9:] in sensorlist:
        coordict_M.update({record['name'][9:]:record['location']['coordinates']})
        print(record['name'][9:], '-', record['location'])
```

```
print()
print('dictionary updated with the coordinates of the low-cost sensors'
    , '-',
    coordict_M)
```

Now, that the updated dictionary contained the coordinates of the low-cost sensors, the mean of these sensors was calculated and added to the dictionary as well. The values of these dictionary were put into a new dataframe. The resulting dataframe consisted of station name - their annual-mean value, latitude, and longitude information. The script for this is mentioned below:

Script 5: Making a dataframe with annual mean, coordinates, and station-id of all the selected sensors

```
# iterating over the low-cost sensor dataframe
for col in df_lowcost:
   # linking the coordinate values and mean values by column(sensor) name
   if col in coordict_M:
      mean = df_lowcost[col].mean()
      coordict_M[col].append(mean)
      collist.append(col)

# creating three lists - latitude, longitude and annual-mean to store it in a dataframe
latlist = []
longlist = []
annualmeanlist = []
for k, v in sorted(coordict_M.items()):
   latlist.append(v[0])
   longlist.append(v[1])
   annualmeanlist.append(v[2])

# creating a dataframe using four values -
# 'station id', 'annual mean', 'latitude' and 'longitude'
df_sub = pd.DataFrame(columns=
                ['station', 'mean_value', 'latitude', 'longitude']
               )
df_sub['station'] = sorted(collist)
df_sub['mean_value'] = annualmeanlist
df_sub['latitude'] = latlist
df_sub['longitude'] = longlist

print(df_sub)
```

The dataframe output at the end of the data analysis process for both networks:

	station	mean_value	latitude	longitude
0	LTD_11395	4.170390	6.502000	53.204000
1	LTD_37833	4.758491	6.589000	53.227000
2	LTD_37974	9.482882	6.640000	53.218000
3	LTD_38008	6.111111	6.575000	53.187000
4	LTD_38243	4.231593	6.574000	53.212000
5	LTD_38343	2.570616	6.523000	53.220000
6	LTD_38352	8.443277	6.594000	53.244000
7	LTD_38923	7.330837	6.506000	53.236000
8	LTD_42141	5.140763	6.580000	53.240000
9	LTD_42437	4.098349	6.546000	53.236000
10	NL10937	8.077289	6.578901	53.217796
11	NL10938	8.564369	6.608937	53.246535

Figure 4: Output Dataframe of Annual Analysis

The above created dataframe was then converted to geopandas dataframe using the script below. This was done to create a shapefile out of the geopandas dataframe that can be used in QGIS to do interpolation or further analysis. *(This step is not necessary as the simple pandas dataframe can also be converted to csv file and then imported in QGIS and it will work just fine for interpolation, but while doing it I thought it would be a better idea to convert to a shapefile to get more flexibility if I wanted to do something more in QGIS or in Python through other geo processing libraries)*

Script 6: Converting the dataframe to a shapefile

```
# setting the geometry using Shapely
from shapely.geometry import Point
df_sub['geometry'] = df_sub.apply(lambda x: Point((float(x.latitude),
                                float(x.longitude))), axis=1
                        )

# converting the pandas dataframe to geopandas dataframe
import geopandas as gpd
gdf = gpd.GeoDataFrame(df_sub, geometry='geometry')

# saving the dataframe to a shapefile
gdf.to_file('PM25.shp', driver='ESRI Shapefile')
```

Thus, at the end of the Annual Analysis process, the outputs were two shapefiles – one for $PM_{2.5}$ and one for NO_2, containing the information about the relevant sensors *(from both 'Lucht-MeetNet' and 'Samen-Meten' network)* such as their coordinates, station-id, and the calculated annual mean values.

Selected Day Analysis: The Selected Day Analysis was performed by keeping in mind the *Question-2* described in the *'Introduction'* Section. Like the *'Annual Analysis'* segment, this process was also divided into two main parts pertaining to the two selected Dutch Air Quality Monitoring Networks. Considering the logic discussed in the *'Annual Analysis'* segment, this segment builds on the same logic but instead of aggregating the filtered values to output an annual mean, the scripts used for this analysis provided hourly values aggregated to 4 different time frame of a selected day – Morning (4 am to 11 am), Noon (11 am – 5 pm), Evening (5 pm – 9 pm), and Night (remaining part of the day). The scripts for both sensor networks are mentioned below:

Script 7: Selected Day Analysis for 'Lucht-MeetNet' Network

```
# library imports
import csv
import pandas as pd
import glob
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# empty list and dictionary to be filled later
collist = []
coordict_D = {}
dailydict = {}

# sensor list containing 'station-id' of interest
sensorlist = ['NL10937', 'NL10938']

for sensors in sensorlist:
    newwlist = []
    newwdict = {}

    # empty mean dictionary to be used later
    meandict = {'mean_value': []}

    # identifying csv files from a specific path
    path = 'PM2.5_CSV/'        # point to the NO2 files to get the NO2 values
    all_files = sorted(glob.glob(path + '/*.csv'))

    month = ['January', 'February', 'March', 'April', 'May', 'June',
             'July', 'August', 'September', 'October', 'November']
    mo = 0

    # list used to transfer values to new dataframe
    transferlist = []

    # iterating over the identified csv files
    for file in all_files:
        f = open(file, encoding='latin-1')
        csv_reader = csv.DictReader(f, delimiter=';')

        # script to put in the elements of the csv file to an empty list
        emptylist = []
        for row in csv_reader:
            emptylist.append(row)

        y = emptylist
        station_id = sensors
        valuelist = []
        datetime = []
```

```
    # script to store the coordinates from the empty list into a dictionary
    for keys, valuess in emptylist[1].items():
        if keys == station_id:
            valuesplit = valuess.split(',')
            coordict_D.update({station_id:
                               [float(valuesplit[1][:-1]),
                                float(valuesplit[0][1:])]
                               }
                              )

    # script to get measured values and time of the station of interest
    for i in range(8, len(emptylist)):
        z = emptylist[i]
        for j in range(len(z)):
            datetime.append(z[''] + ' - ' + z['StationsCode'])
            if station_id in z:
                valuelist.append(z[str(station_id)])
                break

    # storing the above values in a dataframe
    dff = pd.DataFrame(valuelist, columns = ['value'])
    dff['Date_Time'] = datetime
    dff['value'] = pd.to_numeric(dff['value'], downcast='float')

    # calculating the mean of each day and storing it in 'newwlist'
    rowi = 0
    rowj = 23
    meanlist = []
    for index, rows in dff.iterrows():
        if rowj <= len(dff):
            dff_I = dff.loc[range(rowi, rowj)]
            newwlist.append(dff_I['value'].values)
            rowi = rowj
            rowj += 24

    keyDay = 1
    for lists in newwlist:
        newwdict.update({keyDay: lists})
        keyDay += 1
  dailydict.update({station_id: newwdict})

dailydf = pd.DataFrame(dict([(k, pd.Series(v)) for k, v in dailydict.items()]))


def dayextractor(dayNumber):     # function to get daily data by selected Day

  dailyloc = dailydf.loc[[dayNumber]]

  d = dailyloc.to_dict()

  for k, v in d.items():
    print('The minimum recorded measurement today for station', k, 'is',
          min(v[dayNumber]), 'and the time of the measurement is',
          np.argmin(v[dayNumber]), '-', np.argmin(v[dayNumber]) + 1,
          'hour')
    print('The maximum recorded measurement today for station', k, 'is',
          max(v[dayNumber]), 'and the time of the measurement is',
          np.argmax(v[dayNumber]), '-', np.argmax(v[dayNumber]) + 1,
          'hour')
    print()
    listM = []
```

```
        listA = []
        listE = []
        listN = []
        for elem in range(len(v[dayNumber])):
            if elem >= 4 and elem <= 10:
                listM.append(v[dayNumber][elem])
            elif elem > 10 and elem <= 15:
                listA.append(v[dayNumber][elem])
            elif elem > 15 and elem <= 20:
                listE.append(v[dayNumber][elem])
            else:
                listN.append(v[dayNumber][elem])

        meanM = np.mean(np.array(listM))
        meanA = np.mean(np.array(listA))
        meanE = np.mean(np.array(listE))
        meanN = np.mean(np.array(listN))
        d.update({k: [meanM, meanA, meanE, meanN]})

    for k, v in d.items():
        if k in coordict_D:
            attach = 0
            while attach <= 3:
                coordict_D[k].append(d[k][attach])
                attach += 1

    latL = []
    longL = []
    mornL = []
    noonL = []
    evenL = []
    nightL = []
    stationL = []
    for k, v in coordict_D.items():
        stationL.append(k)
        latL.append(v[0])
        longL.append(v[1])
        mornL.append(v[2])
        noonL.append(v[3])
        evenL.append(v[4])
        nightL.append(v[5])

    dailyplotdf = pd.DataFrame(stationL, columns = ['station_id'])
    dailyplotdf['latitude'] = latL
    dailyplotdf['longitude'] = longL
    dailyplotdf['morning'] = mornL
    dailyplotdf['noon'] = noonL
    dailyplotdf['evening'] = evenL
    dailyplotdf['night'] = nightL

    print(dailyplotdf)
    print()
    print('process finished! - run next block')

dayextractor(100)
```

Script 8: Selected Day Analysis for 'Samen-Meten' Network

```
# library imports
import urllib3
import json
import csv
import pandas as pd
import glob
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# empty mean dictionary to be used later
meandict = {'mean_value':[]}

# identifying csv files from a specific path
path = 'MTPM25CSV/'        # point to the NO2 files to get the NO2 values
all_files = sorted(glob.glob(path + '/*.csv'))

# storing the name (id) of the sensor in a list
sensorlist = []

newtransferlist = []
for filename in all_files:
    namesplit = filename.split('/')
    sensorlist.append(namesplit[1][0:9])

# iterating over the csv files
iterr = 0
for file in all_files:
    f = open(file, encoding='latin-1')
    csv_reader = csv.DictReader(f, delimiter=';')

    # taking all the values of a csv file and storing it to a list
    emptydict = {}
    dflist = []
    datelist = []
    for row in csv_reader:
        emptydict.update(row)
        for k, v in emptydict.items():
            strsplit = v.split(',')
            if strsplit[1] == '':
                strsplit[1] = 0
                dflist.append(float(strsplit[1]))
            else:
                dflist.append(float(strsplit[1]))
            datelist.append(strsplit[0])

    # appending the list with measured values to a new transfer list
    # this list will be converted into a dataframe
    newtransferlist.append(dflist)

# finding the list with maximum length amongst the other list
# this deals with the problem of difference in data availability for sensors
list_length = [len(dflist) for dflist in newtransferlist]
link_position = np.argmax(np.array(list_length))

# linking the value list with the sensor list and storing it to a dataframe
for lp in range(len(newtransferlist)):
    if lp == link_position:
        df_lowcost = pd.DataFrame({sensorlist[lp]: pd.Series(newtransferlist[lp])})
```

```
for lp in range(len(newtransferlist)):
    if lp != link_position:
        df_lowcost[sensorlist[lp]] = pd.Series(newtransferlist[lp])

df_lowcost = df_lowcost.reindex(sorted(df_lowcost.columns), axis=1)

http = urllib3.PoolManager()

# creating a custom url to filter out locations in a particular polygon area
url = "https://api-samenmeten.rivm.nl/v1.0/Locations?$filter=st_intersects(location, "
url += "geography'POLYGON((6.646072 53.245780, 6.677191 53.185053, 6.641361 53.159123, "
url += "6.561241 53.146227, 6.492015 53.157271, 6.412582 53.166608, 6.367357 53.200069, "
url += "6.344730 53.237277, 6.376067 53.281637, 6.567162 53.287902, 6.719266 53.273015, "
url += "6.777630 53.244263, 6.646072 53.245780))')"

request = http.request('GET', url)

data = json.loads(request.data.decode('utf-8'))

# iterating over the acquired data
for record in data['value']:
    # filtering only relevant sensors from all records and updating the coordinate dictionary
    if record['name'][9:] in sensorlist:
        coordict_D.update({record['name'][9:]:record['location']['coordinates']})

dicti = df_lowcost.to_dict()


def dayextractornew(dayNumber):    # function to get daily data by selected Day

    rowi = 0 + (dayNumber - 1)*23
    rowj = 23 + (dayNumber - 1)*23

    for key, value in dicti.items():
        twolist = list(value.values())[rowi:rowj]
        dicti.update({key: twolist})

    for kee, val in dicti.items():
        listM = []
        listA = []
        listE = []
        listN = []
        for iterr in range(len(val)):
            if iterr >= 4 and iterr <= 10:
                listM.append(val[iterr])
            elif iterr > 10 and iterr <= 15:
                listA.append(val[iterr])
            elif iterr > 15 and iterr <= 20:
                listE.append(val[iterr])
            else:
                listN.append(val[iterr])

        meanM = np.mean(np.array(listM))
        meanA = np.mean(np.array(listA))
        meanE = np.mean(np.array(listE))
        meanN = np.mean(np.array(listN))
        dicti.update({kee: [meanM, meanA, meanE, meanN]})

    for key, value in dicti.items():
        if key in coordict_D:
```

```
        attach = 0
        while attach <= 3:
          coordict_D[key].append(dicti[key][attach])
          attach += 1

  latL = []
  longL = []
  mornL = []
  noonL = []
  evenL = []
  nightL = []
  stationL = []
  for k, v in coordict_D.items():
    stationL.append(k)
    latL.append(v[0])
    longL.append(v[1])
    mornL.append(v[2])
    noonL.append(v[3])
    evenL.append(v[4])
    nightL.append(v[5])

  dailyplotdf = pd.DataFrame(stationL, columns = ['station_id'])
  dailyplotdf['latitude'] = latL
  dailyplotdf['longitude'] = longL
  dailyplotdf['morning'] = mornL
  dailyplotdf['noon'] = noonL
  dailyplotdf['evening'] = evenL
  dailyplotdf['night'] = nightL

  print(dailyplotdf)
  dailyplotdf.to_csv('selected_day.csv')

dayextractornew(100)
```

(Note: As seen in the above scripts, the number of the day of the year 2021 was selected to be 100, which was 10th April 2021. However, to reproduce or analyze data for some other day, the day number can be changed in the function parameter and the scripts will output data for that selected day.)

The output dataframe (which was also saved to .csv file for further analysis):

	station_id	latitude	longitude	morning	noon	evening	night
0	NL10937	6.578901	53.217796	3.049143	2.2616	0.8296	2.818571
1	NL10938	6.608937	53.246535	4.462714	4.2954	4.2762	3.959143
2	LTD_42437	6.546000	53.236000	1.857143	2.2000	2.4000	2.500000
3	LTD_42141	6.580000	53.240000	5.428571	6.6000	8.8000	7.333333
4	LTD_38243	6.574000	53.212000	1.857143	1.0000	2.4000	1.333333
5	LTD_38923	6.506000	53.236000	2.857143	2.2000	4.6000	3.166667
6	LTD_38352	6.594000	53.244000	4.000000	2.4000	4.0000	3.166667
7	LTD_38343	6.523000	53.220000	0.857143	1.8000	1.2000	1.166667
8	LTD_38008	6.575000	53.187000	2.571429	2.6000	2.4000	2.500000
9	LTD_37974	6.640000	53.218000	4.714286	4.0000	4.4000	3.166667
10	LTD_37833	6.589000	53.227000	0.285714	0.6000	1.0000	0.000000
11	LTD_11395	6.502000	53.204000	10.571429	6.0000	3.2000	2.500000

Figure 5: Output Dataframe of Selected Day Analysis

Thus, by the end of the Selected Day Analysis process, the output was a .csv file with the format of the data similar to the format of data in the output dataframe ***(Figure 5).***

QGIS Processing: For the next part of the Data Analysis, the shapefiles and csv files generated during the *'Annual Analysis'* and *'Selected Day Analysis'* segments were imported in QGIS to answer the defined questions and produce maps that can be useful to both the local government and the citizens to make informed decisions.

- ***District Analysis:*** For district analysis of air quality and answering the defined *Question-1*, the shapefiles produced during Annual Analysis segment were imported to QGIS. Additionally, other data such as Provinces and District datasets were downloaded from PDOK. Using these datasets, the city area of Groningen was filtered out. The resulting layer was then used to clip the district dataset to get the districts of Groningen. Further, the 'mean_value' field from the $PM_{2.5}$ and NO_2 shapefiles was interpolated using Inverse Distance Weighted (IDW) Algorithm of QGIS by selecting the extracted Groningen city area as extent of interpolation. The result of this process was two Interpolated raster layers ($PM_{2.5}$ and NO_2). These layers were then clipped using the districts of Groningen. The results were two interpolated maps ($PM_{2.5}$ and NO_2) containing information about the annual mean values of these components in particular districts of Groningen. The maps and the values calculated using 'Zonal Statistics' are shown in the 'Results' section.

- ***Recreational Area Analysis:*** To analyze the values of these components surrounding the recreational areas throughout the day (defined *Question-2*), similar approach was followed as District Analysis. First, Land-Use dataset was downloaded from PDOK. This dataset along with the csv files produced during Selected Day Analysis segment were imported to QGIS. The Land-Use dataset was clipped using the extent of Groningen city area created during District Analysis. Further, the values of 'Recreatie' (Dutch for Recreational) were filtered out from the clipped layer. The result of this process was a vector layer containing the recreational areas of Groningen. As the '*Question'* asks about the air quality 'in' and 'around' the recreational areas, a 200m buffer was created surrounding the recreational areas. Using the area of Groningen as extent, the csv files were interpolated using the values 'Morning', 'Noon', 'Evening', and 'Night' fields. The resulting interpolated raster was clipped by the buffered recreational areas and statistics were calculated to find the mean values for selected time-period of the day. Thus, at the end of this process, the resulting outputs were 4 maps containing information about the air quality values for a selected during Morning (4 am to 11 am), Noon (11 am – 5 pm), Evening (5 pm – 9 pm), and Night (remaining part of the day). The maps and the values calculated using 'Zonal Statistics' and 'Basic Statistics for Fields' are shown in the 'Results' section.

- ***Running Path Analysis:*** To analyze the best street/running paths in Groningen, first the entire area of Groningen was divided into three parts: North-East, South, and Northwest-Center and related areas were extracted. Then, a 10m buffer was created around the road network layer to convert it to polygon and facilitate further operations. Further, the buffered road layer was clipped using the extracted three sub areas as overlay. Both $PM_{2.5}$ and NO_2 interpolated rasters were clipped using the resulting road network layer from the previous operation as extent. Thus, the results contained interpolated pixel values along the road network for each of the three areas. Then, using the 'Zonal Statistics' tool, the best street/running path were noted. This answers the defined *Question-3*, however, there's a limitation to this procedure. For instance, since the area of Groningen was only divided into three sub-areas, there's a possibility that a person can live far from the best running street. Thus, to avoid this limitation, the maps were produced in a way that one can easily look at the map and identify the closest best street to one's location.

3. Results

Daily Air-Quality Values in Groningen:

Since the data used had quantitative characteristics and was analyzed over a timeframe, making a calendar heatmap was a good option to display this quantitative data over a time series. As a calendar heatmap combines calendar view and heatmap, it makes it a lot easy to make sense of the presented values and identify the trend that the values are following over a time series. These calendar heatmaps produced would help the end-user (government or citizens) gain more insight and information about the air-quality in Groningen.

- *Calendar Heatmaps produced during **Script 2** for daily measured $PM_{2.5}$ values ($\mu g/m^3$) 'Lucht-MeetNet' Sensors of Groningen*

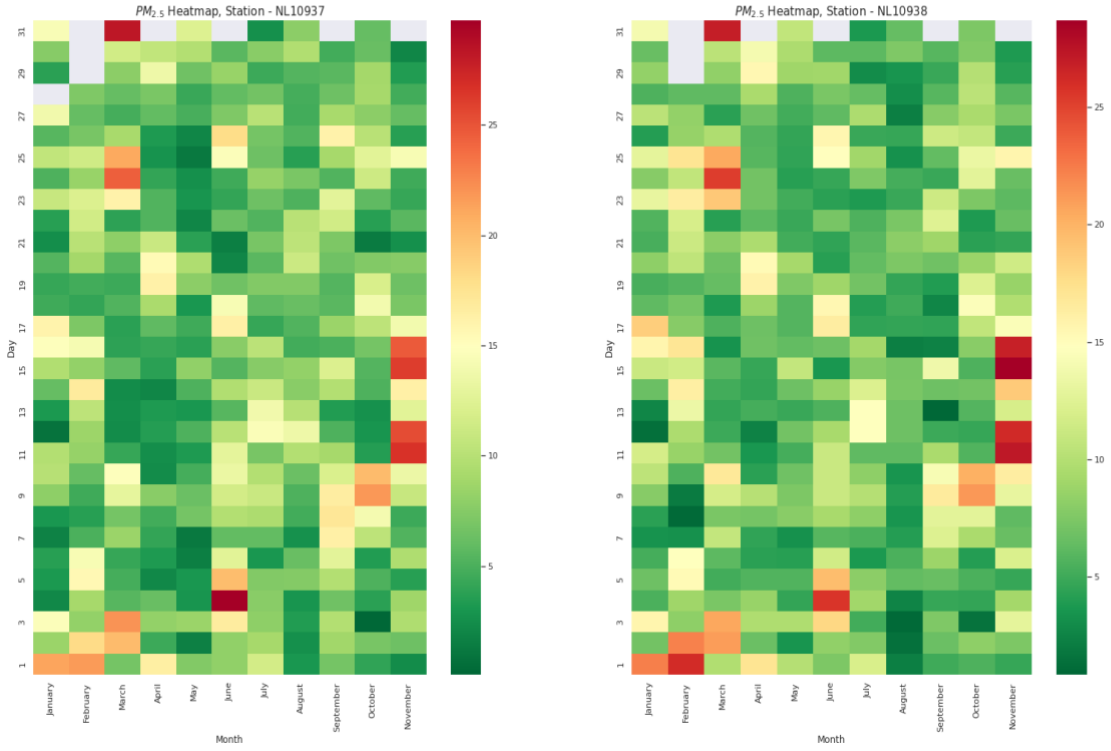


Figure 6: Calendar Heatmaps for daily values of $PM_{2.5}$ ($\mu g/m^3$) – (WHO Standards Limit - 25$\mu g/m^3$ per day)

As seen via the calendar heatmaps above there are some days, for which the $PM_{2.5}$ values exceed that of the defined WHO Standards.

- *Calendar Heatmaps produced during **Script 2** for daily measured NO_2 values ($\mu g/m^3$) 'Lucht-MeetNet' Sensors of Groningen*

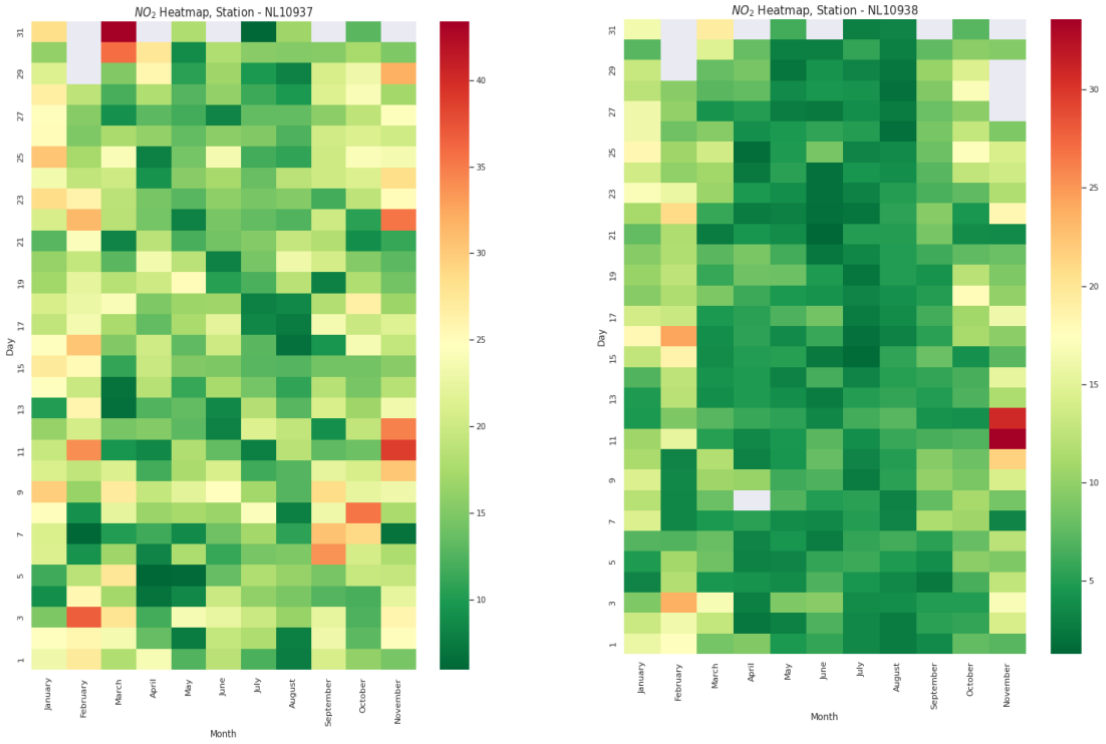


Figure 7: Calendar Heatmaps for daily values of NO_2 ($\mu g/m^3$) – (No defined Daily Standards)

Air-Quality in various Districts of Groningen:

Below displayed two maps show the $PM_{2.5}$ and NO_2 values in various districts of Groningen. Along with these maps, there are also two tables containing the measured annual mean values of the selected components in various districts of Groningen. These results help us address the defined *Question-1*.

- *Map for Annual Mean (January 2021 – November 2021) $PM_{2.5}$*

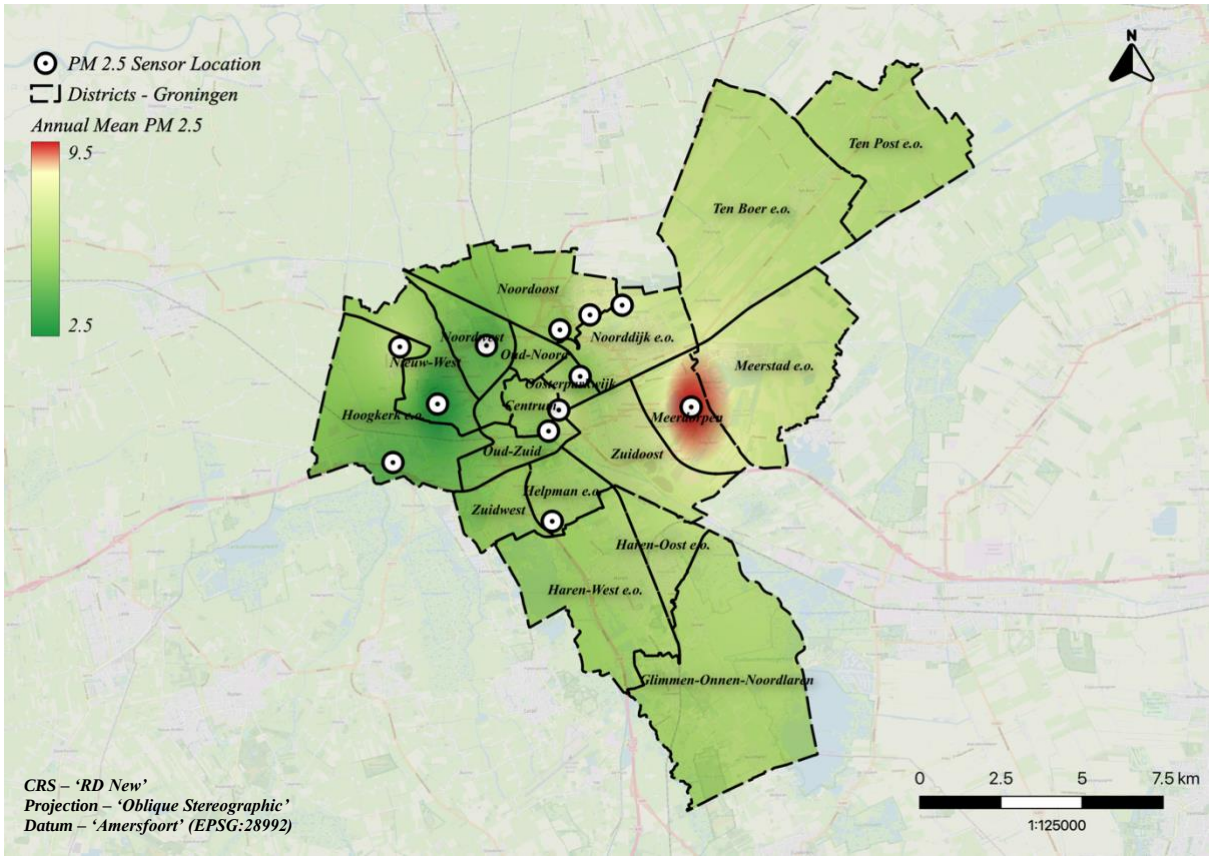


Figure 8: Map of Distribution of annual mean $PM_{2.5}$ ($\mu g/m^3$) in various districts of Groningen

- *Table for Annual Mean (January 2021 – November 2021) $PM_{2.5}$*

District	***Annual Mean $PM_{2.5}$ ($\mu g/m^3$)***
Centrum	5.72
Glimmen-Onnen-Noordlaren	6.62
Haren-Oost e.o.	6.82
Haren-West e.o.	6.16
Helpman e.o.	5.92
Hoogkerk e.o.	5.07
Meerdorpen	8.53
Meerstad e.o.	7.65
Nieuw-West	4.86
Noorddijk e.o.	7.46
Noordoost	5.95
Noordwest	4.67

Oosterparkwijk	5.87
Oud-Noord	5.59
Oud-West	4.94
Oud-Zuid	5.25
Ten Boer e.o.	7.14
Ten Post e.o.	6.75
Zuidoost	6.92
Zuidwest	5.47

Table 2: Annual Mean Values of $PM_{2.5}$ for the time frame January 2021 – November 2021 in various districts of Groningen

- *Map for Annual Mean (January 2021 – November 2021) NO_2*

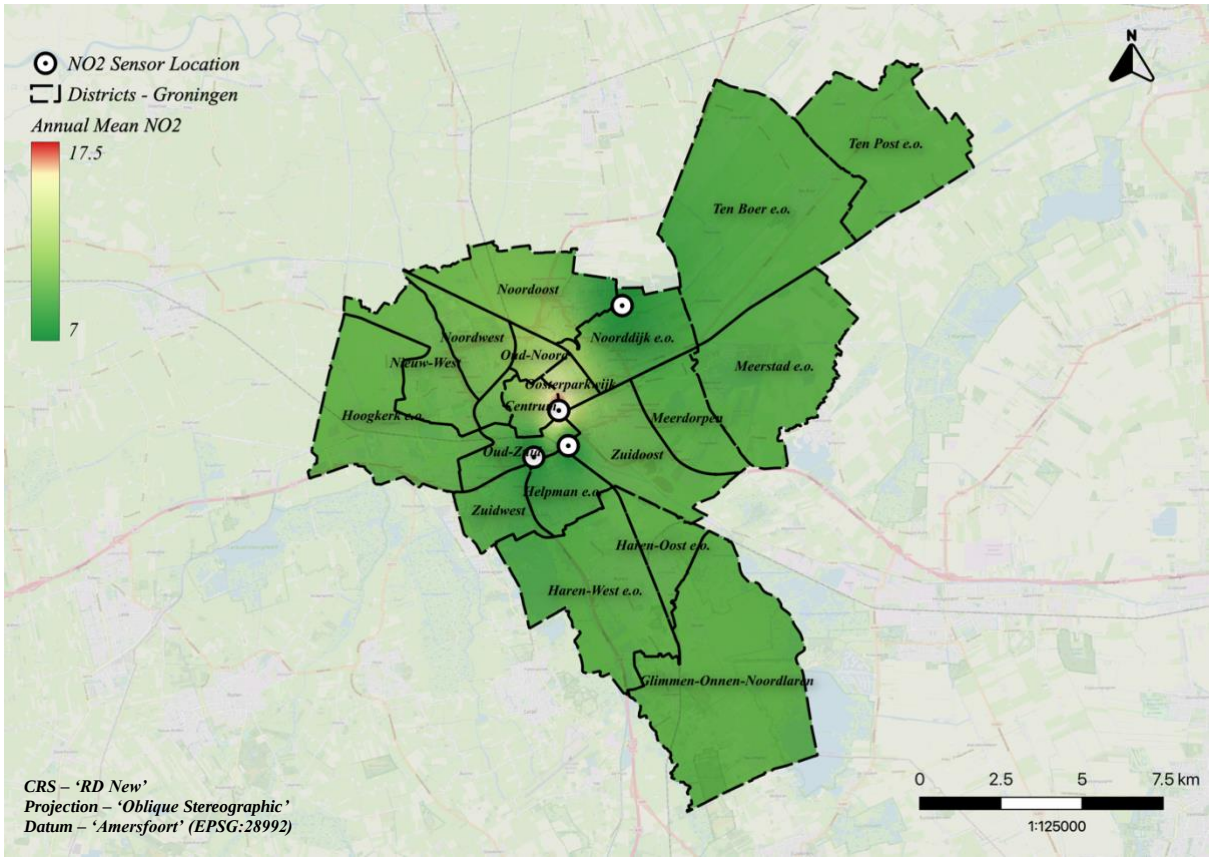


Figure 9: Map of Distribution of annual mean NO_2 ($\mu g/m^3$) in various districts of Groningen

- *Table for Annual Mean (January 2021 – November 2021) NO_2*

District	***Annual Mean NO_2 ($\mu g/m^3$)***
Centrum	12.05
Glimmen-Onnen-Noordlaren	9.80
Haren-Oost e.o.	9.79
Haren-West e.o.	9.60
Helpman e.o.	8.77
Hoogkerk e.o.	10.01
Meerdorpen	9.48
Meerstad e.o.	9.50
Nieuw-West	10.16
Noorddijk e.o.	8.97
Noordoost	10.38
Noordwest	10.57
Oosterparkwijk	13.82
Oud-Noord	12.20
Oud-West	10.22
Oud-Zuid	9.45
Ten Boer e.o.	9.31
Ten Post e.o.	9.63
Zuidoost	9.93
Zuidwest	9.11

Table 3: Annual Mean Values of NO_2 for the time frame January 2021 – November 2021 in various districts of Groningen

Keeping the defined air-quality standards in mind and observing the above results, one can interpret that no district in Groningen has an annual mean value *(for both $PM_{2.5}$ and NO_2)* that exceeds the standards defined by WHO and EEA. Furthermore, these maps can be taken into consideration by the local government to make decisions such as setting up industry, corporate offices, parks, recreational spaces, etc. which all involves air-quality assessment.

Air-Quality in and around Recreational Spaces in Groningen:

Below displayed maps show how the measured values of the defined air-quality components vary throughout a selected day in and around recreational areas in Groningen. These results help us address the defined *Question-2*.

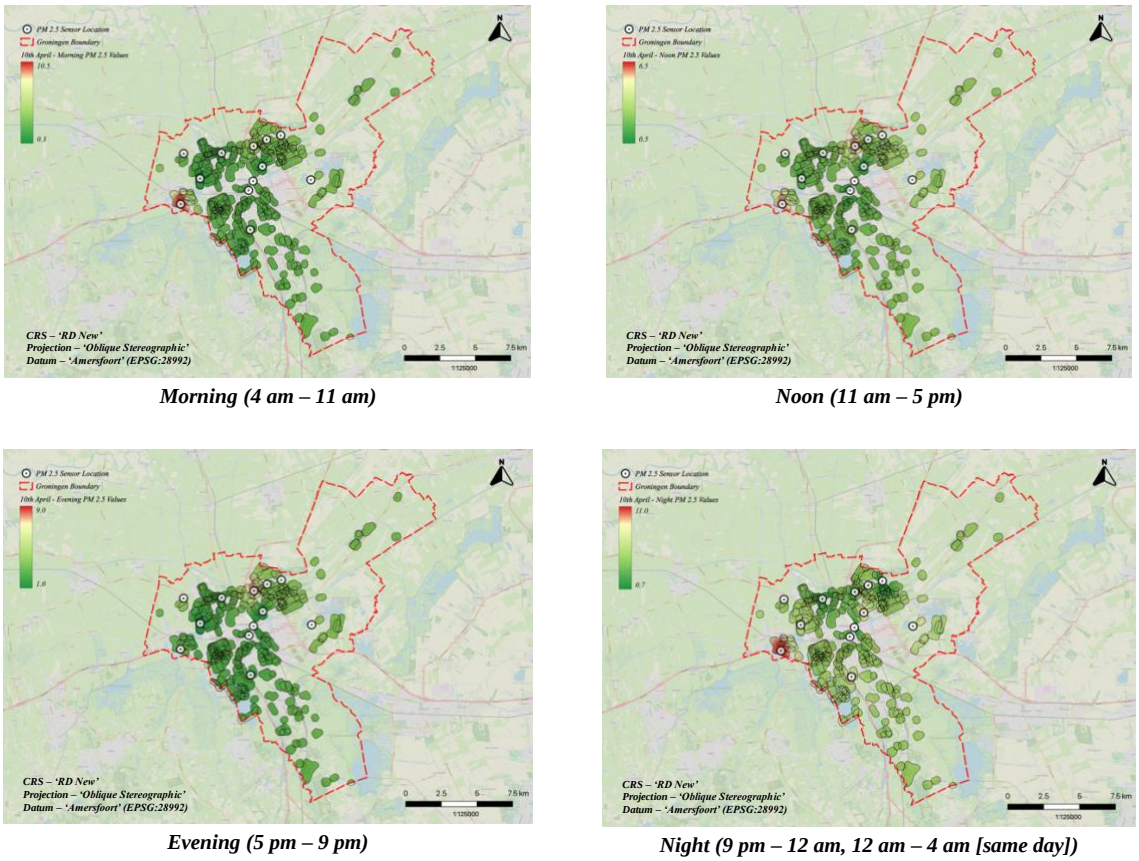


Figure 10: Maps showing the variation in the measured $PM_{2.5}$ values throughout the day of 10th April 2021

As observed through above maps, the dark green color *(low values of $PM_{2.5}$)* is decreasing throughout the day. However, it's hard to tell the difference in some regions. Thus, below table provides the calculated mean of $PM_{2.5}$ for each time-period of the day in and around 200m buffer of the recreational spaces.

Time of the Day	*$PM_{2.5}$ Value ($\mu g/m^3$)*
Morning (4 am – 11 am)	3.27
Noon (11 am – 5 pm)	2.90
Evening (5 pm – 9 pm)	3.19
Night (9 pm – 12 am, 12 am – 4 am [same day])	5.36

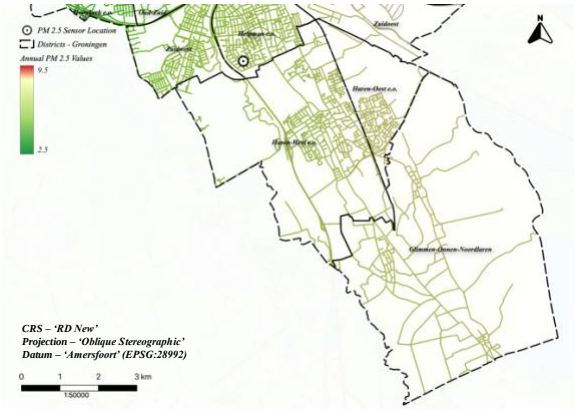
Table 4: Values of $PM_{2.5}$ ($\mu g/m^3$) throughout the day of 10th April 2021 in and around recreational areas

Thus, supplementing the visual perception of the maps, the above table also suggests that value of $PM_{2.5}$ is in-fact increasing throughout the day *(decrease in dark green color)*. However, noon and evening have slightly better values than morning. This can help local people make better decisions regarding the appropriate time *(in context of air-quality)* to do outdoor exercises.

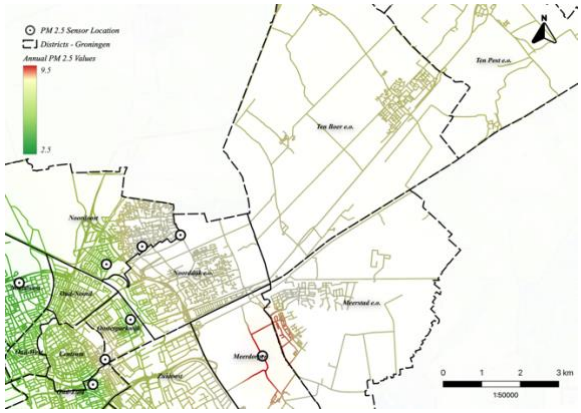
Air-Quality surrounding the Road Network in Groningen:

Below maps show the air-quality along the road networks in the three selected sub-areas of Groningen

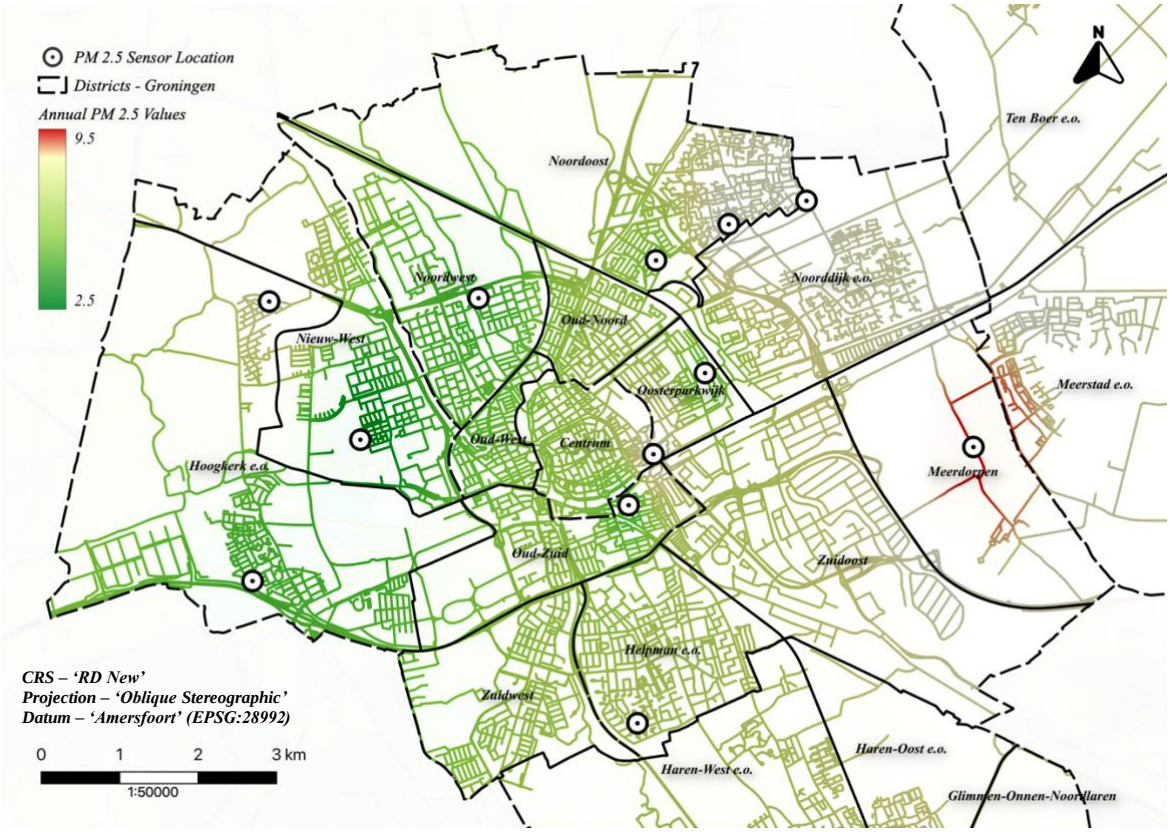
- *Maps for Annual Mean $PM_{2.5}$ Values along road networks of Groningen*



South Sub-Area of Groningen



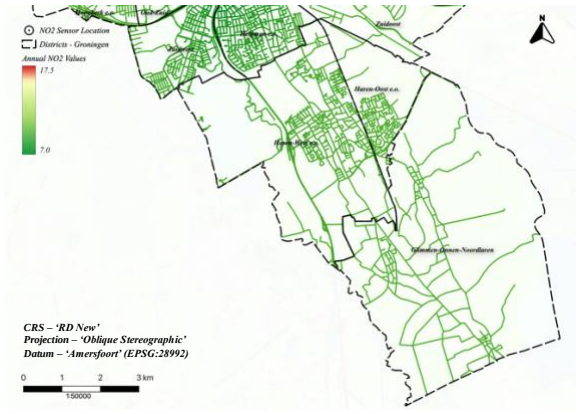
North-East Sub-Area of Groningen



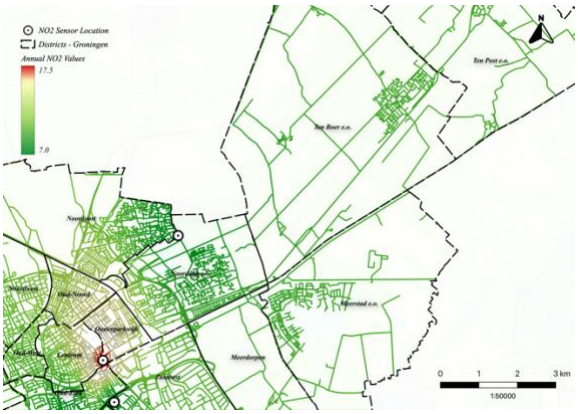
Northwest-Center Sub-Area of Groningen

Figure 11: Maps showing the annual mean $PM_{2.5}$ ($\mu g/m^3$) values along the road networks in Groningen

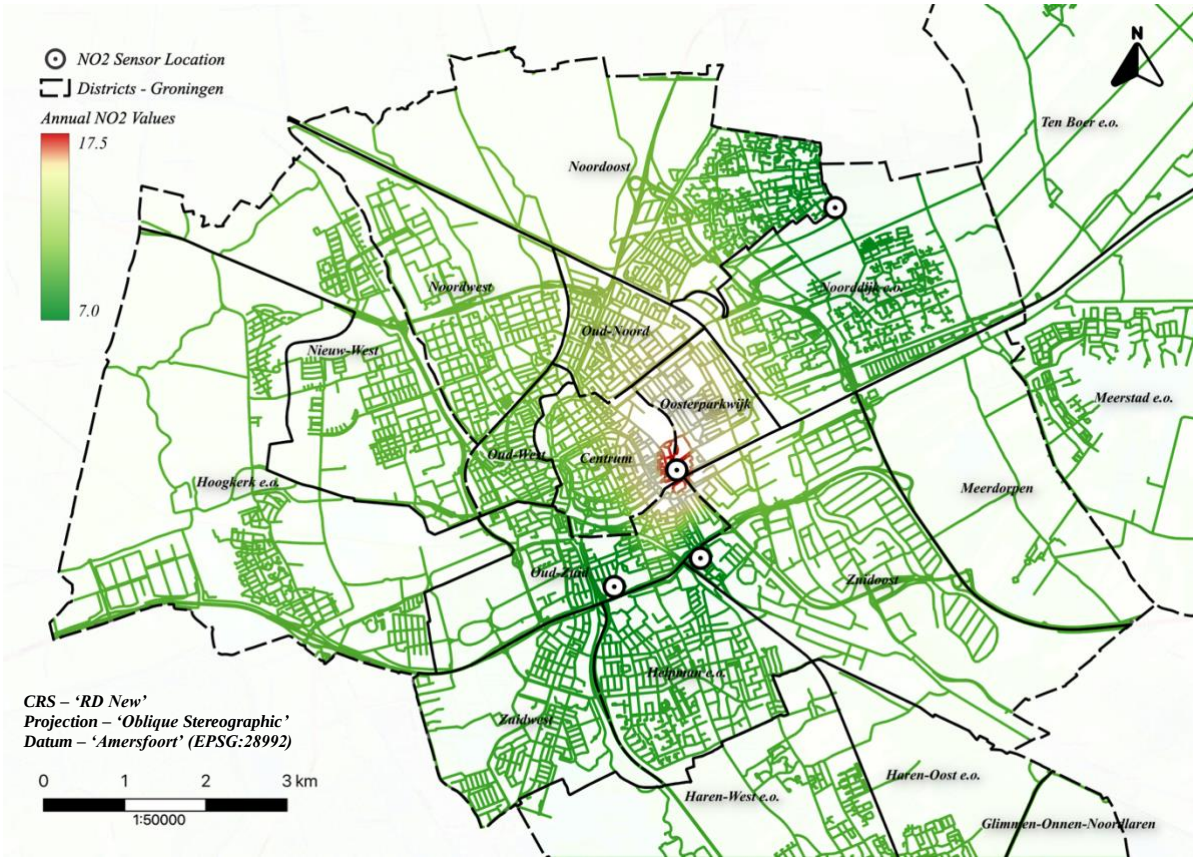
- *Maps for Annual Mean NO_2 Values along road networks of Groningen*



South Sub-Area of Groningen



North-East Sub-Area of Groningen



Northwest-Center Sub-Area of Groningen

Figure 12: Maps showing the annual mean NO_2 ($\mu g/m^3$) values along the road networks in Groningen

The above map can help individuals make better decision regarding the best street that they can use for running/jogging based on an individual's location of residence. However, since we know from the previous results that overall the annual mean value of the components doesn't exceed the defined air-quality standards, one can choose any street. But, to answer the defined *Question-3* precisely, the table below shows the name of the streets with the best air-quality *(low levels of $PM_{2.5}$ and NO_2)* for the above selected three sub-areas of Groningen.

Sub-Area of Groningen	***Name of the Street***	***Annual $PM_{2.5}$ Value ($\mu g/m3$)***
Northwest-Center	Onyxstraat	2.57
South	Winterpad	5.57
	Veenweg	5.59
North-East	Oostersluisweg	5.07

Table 5: Streets with lowest $PM_{2.5}$ ($\mu g/m^3$) values in the defined sub-areas of Groningen

Sub-Area of Groningen	***Name of the Street***	***Annual NO_2 Value ($\mu g/m^3$)***
Northwest-Center	Vredelust	6.91
	De Zaayer	7.01
	Verlengde Meeuwerderweg	7.00
South	Boumaboulevard	7.81
North-East	Kardingermaar	8.70
	Stadsweg	8.81

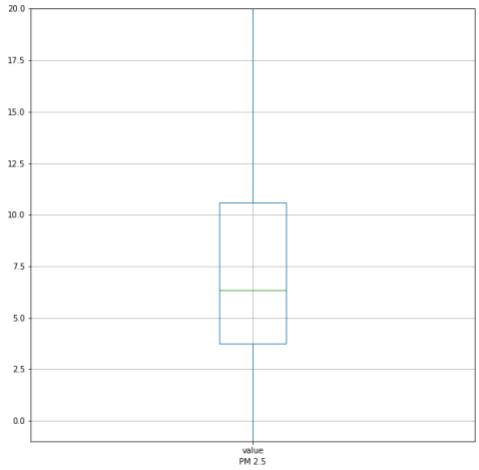
Table 6: Streets with lowest NO_2 ($\mu g/m^3$) values in the defined sub-areas of Groningen

4. Discussion

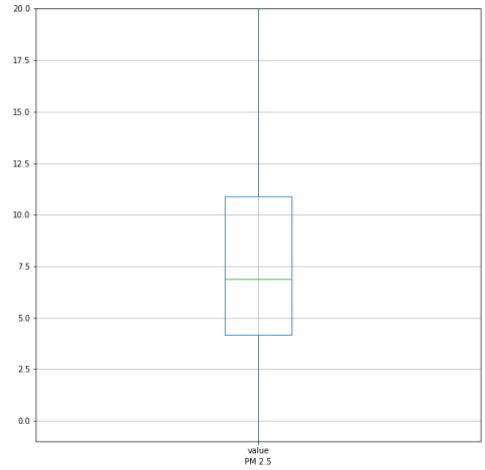
Data Quality and Assessment:

Before discussing the validation of results, it is important to assess the quality of the data used. The data obtained from the '*Lucht-MeetNet*' network is high-quality as it is maintained by the government of the Netherlands. But there are only two working sensors of this network that are available in the city area of Groningen. Since, the local government has a huge role to play in setting up the sensors and maintaining them, there is a possibility that the budget doesn't allow the local government to put more high-quality sensors. Another viewpoint is that if a particular area is meeting up to the air-quality standards then that area wouldn't require much attention and thus less sensors. And as we can see in the results, the city area of Groningen does indeed meet these defined standards but then again to continue meeting up to those standards, one also needs to keep on measuring more.

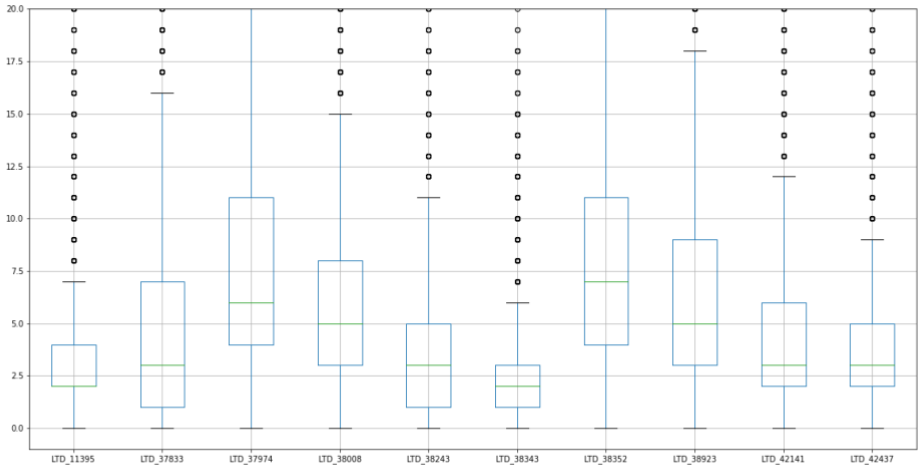
Taking this into account, RIVM has also set up a low-cost sensor network *('Samen-Meten')*. As one would imagine, low-cost sensors are not the best quality sensors that one can get. Also, maintenance of the data for such a network could be questionable. However, comparing the boxplots of 10 selected sensors from the '*Samen*-Meten' network and the 2 main sensors from the '*Lucht-MeetNet*' network, we can see that the median, Q1 *(25^{th} Quartile)*, and Q3 *(75^{th} Quartile)* values are in the same range. Thus, if we consider the '*Lucht-MeetNet*' network data to be of high-quality then the values measured by the low-cost sensors should be acceptable as well as they are in the same range. However, as one can see from the boxplots there are a lot of outliers in the measured values of low-cost sensors. Typically, these outliers are related to the errors in measurement. These errors can be caused due to wide variety of reasons – bad positioning of the sensors, low calibration, and random noise spikes. Despite this, as it can be inferred from above the range of the values is consistent when we consider a yearly timeframe, thus, we can say the outliers wouldn't impact the results much when considering this timeframe. With this, we can conclude the data used for the analysis from the selected sensors has good quality.



NL_10937 station $PM_{2.5}$ ($\mu g/m^3$) – 'Lucht-MeetNet' Network



NL_10938 station $PM_{2.5}$ ($\mu g/m^3$) – 'Lucht-MeetNet' Network



'Samen-Meten' Sensor Network $PM_{2.5}$ ($\mu g/m^3$) values

Figure 13: Boxplots of 'Lucht-MeetNet' Network and 'Samen-Meten' Network for $PM_{2.5}$ ($\mu g/m^3$)

Focusing on the component of NO_2, there were only a total of 4 sensors *(2 from 'Lucht-MeetNet' Network and 2 from 'Samen-Meten' Network)* set up which were measuring the NO_2 values for the city of Groningen. Considering the spatial context, one can argue that such a low number of sensors are not enough to describe the story of NO_2 in the entire city. On the other hand, one can also argue that one of the basic sources of both $PM_{2.5}$ and NO_2 components is vehicular pollution. Thus, there is some sort of correlation between these components and the trend of one can be determined by the other. To assess this, two correlation plots were made for both '*Lucht-MeetNet'* network sensors since they measure both $PM_{2.5}$ and NO_2. The correlation coefficient was found to be 0.25 for station '*NL10937'* and 0.32 for station 'NL10938'. This suggests low correlation between these values. Based on this, we can argue that *(even though the air-quality doesn't exceed the defined standards)* more sensors should be added to the sensor network to cover the area of the city more accurately.

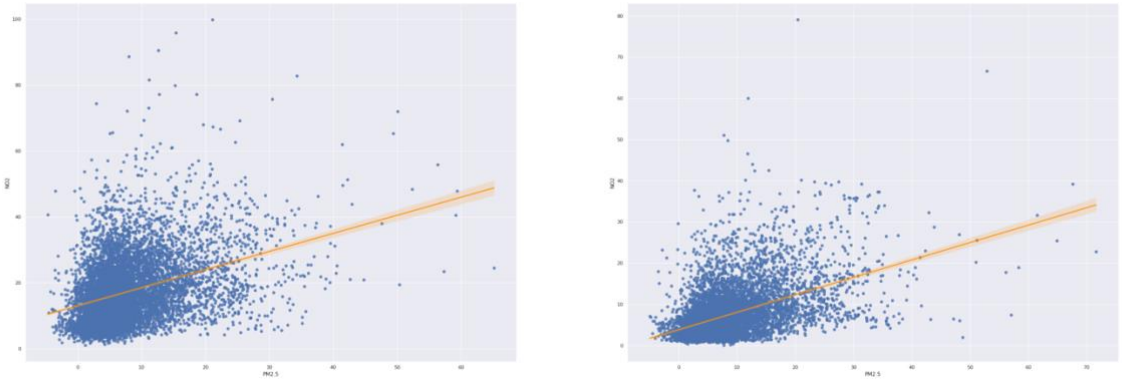


Figure 14: Scatterplot between $PM_{2.5}$ ($\mu g/m^3$) and NO_2 ($\mu g/m^3$)

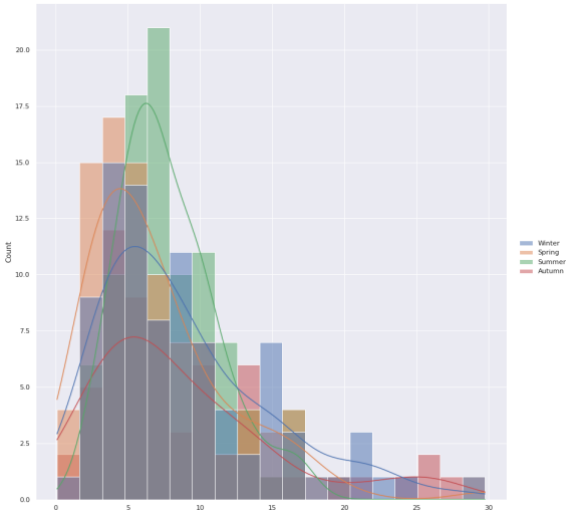
(Note – All the plots seen throughout the report were made while arranging the data into the dataframe using Python)

Validation of Results:

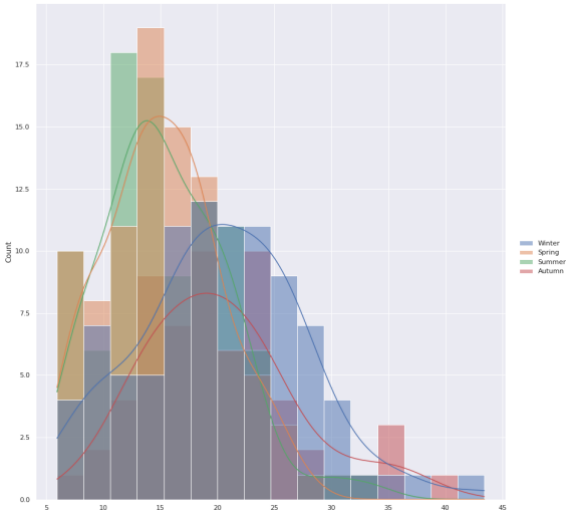
Since, the entire data analysis was done through python, there was a low margin for error as the downloaded data was not cleaned/arranged manually. Also, values produced were in expected range considering the air-quality values displayed in the Groningen area by various portals. Apart from this, the quality of the data used was also good (as assessed in the above segment). Thus, related to this data, the results produced should also be accurate. Considering the defined *Question-2*, ***Table 4*** shows that the $PM_{2.5}$ values measured during morning and night are higher than noon and evening values. This observation was found to be consistent with a study conducted in China (Ni et al., 2019), where it was found that 25 provinces in China had low $PM_{2.5}$ concentrations in the evening as compared to morning.

Reproducibility:

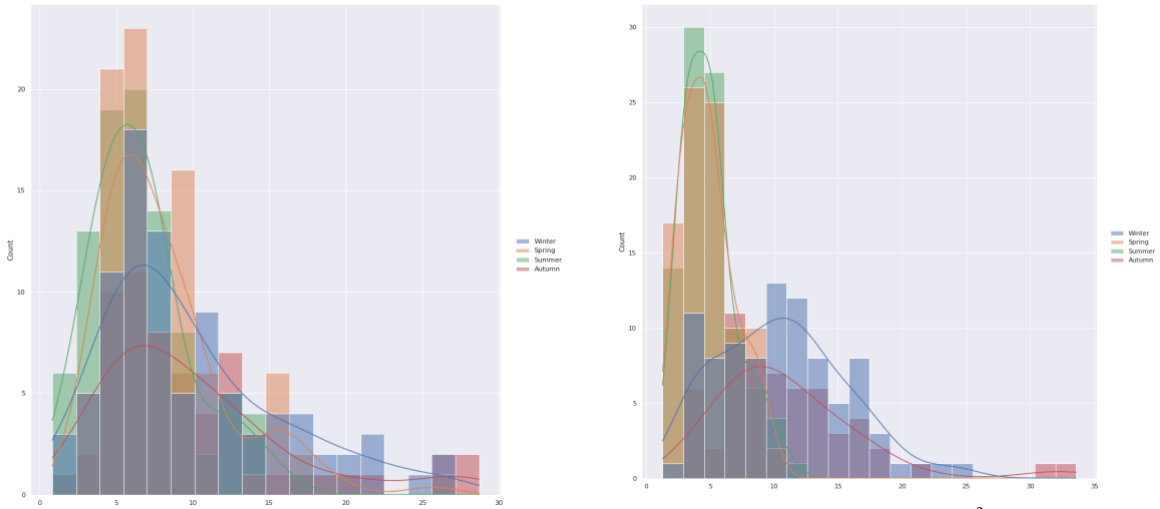
The scripts presented with the report are robust enough to work for any sensor/collection of sensors for any area in the Netherlands. Related to the ***Selected Day Analysis***, a different day and the sensor-id can be given as an input to get to get the aggregated morning, noon, evening, and night values, which can be useful as well. As everything was converted to pandas dataframe, it can easily facilitate different timeframe analysis. For instance, the seasonal variation of $PM_{2.5}$ *(beyond the scope of this project)* is shown below:



NL_10937 $PM_{2.5}$ ($\mu g/m^3$)



NL_10937 NO_2 ($\mu g/m^3$)



NL_10938 $PM_{2.5}$ ($\mu g/m^3$) *NL_10938 NO_2 ($\mu g/m^3$)*

Figure 15: Seasonal Distribution Plots of $PM_{2.5}$ ($\mu g/m^3$) and NO_2 ($\mu g/m^3$) for different sensors

Thus, overall, the scripts developed, and the methodology followed during the analysis can be useful to extend the research to some other timeframe or to repeat the research for the same timeframe but in the coming years.

5. Conclusion

Based on the results of this analysis, we can conclude that, considering the components $PM_{2.5}$ and NO_2 for the city area of Groningen, the annual mean values of these components, as observed, doesn't exceed the limits set by WHO and EEA. However, to maintain these standards, an effort needs to be made to add more sensors to the sensor network as the correlation of the measured values between these components was found to be low, so there can be a problem if the trend is generalized. Relating to the analysis of a selected day, we observed a trend which showed us that evening $PM_{2.5}$ values were low as compared to morning $PM_{2.5}$ around recreational areas. However, the difference wasn't much. To assist local government and citizens to make better decisions and to sum it all up, following conclusions are evident:

The district with the lowest annual-mean $PM_{2.5}$ value	***Noordwest***
The district with the highest annual-mean $PM_{2.5}$ value	***Meerdorpen***
The district with the lowest annual-mean NO_2 value	***Helpman e.o.***
The district with the highest annual-mean NO_2 value	***Oosterparkwijk***
The street with the lowest annual-mean $PM_{2.5}$ value	***Onyxstraat***
The street with the lowest annual-mean NO_2 value	***Vredelust***

Table 7: Concluding Table

6. References

- WHO global air quality guidelines. Particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide. Geneva: World Health Organization; 2021. Licence: CCBY -NC-SA 3.0 IGO.
- Nathanson, J. A. (2020, October 19). air pollution. Encyclopedia Britannica. https://www.britannica.com/science/air-pollution
- Ni, X. F., Peng, S. C., & Wang, J. Z. (2019). Is morning or evening better for outdoor exercise? An evaluation based on nationwide PM2.5 data in China. *Aerosol and Air Quality Research*, *19*(9), 2093–2099. https://doi.org/10.4209/aaqr.2019.07.0362