Air Quality Analysis and Prediction in Tamil Nadu

Phase 4

23-10-2023

Problem Definition: The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels. This project involves defining objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and relevant libraries.

Exploratory Data Analysis & Visualizations

SO_2 **(Sulfur Dioxide)**:

- Colorless gas with a pungent odor.
- It is produced by burning fossil fuels containing sulfur, such as coal and oil.
- It can lead to respiratory problems and contribute to the formation of acid rain.

NO_2 **(Nitrogen Dioxide)**:

- Reddish-brown gas that forms when nitrogen oxides (NOx) react with the atmosphere.
- It is primarily released from combustion processes in vehicles and industrial activities.
- It contribute to respiratory issues and the formation of ground-level ozone, which is harmful to health.

RSPM/PM10 (Respirable Suspended Particulate Matter/Particulate Matter with a diameter of 10 micrometers or less):

- Tiny solid or liquid particles suspended in the air with a diameter of 10 micrometers or less.
- These particles can come from various sources, including dust, vehicle emissions, construction activities, and industrial processes.
- RSPM/PM10 can be inhaled into the lungs, leading to respiratory problems and other health issues.

UNIT OF MEASUREMENT: $\mu g/m^3$ (**micrograms per cubic meter**)

Imports

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
from collections import defaultdict
import folium
import math
import random
from folium.plugins import HeatMapWithTime
from matplotlib.dates import date2num
from matplotlib.lines import Line2D
from folium.map import Layer
from jinja2.environment import Template
```

```
# Color Scheme
so2_color = "#EBCB8B"
no2_color = "#BF616A"
pm10_color = "#4C566A"
```

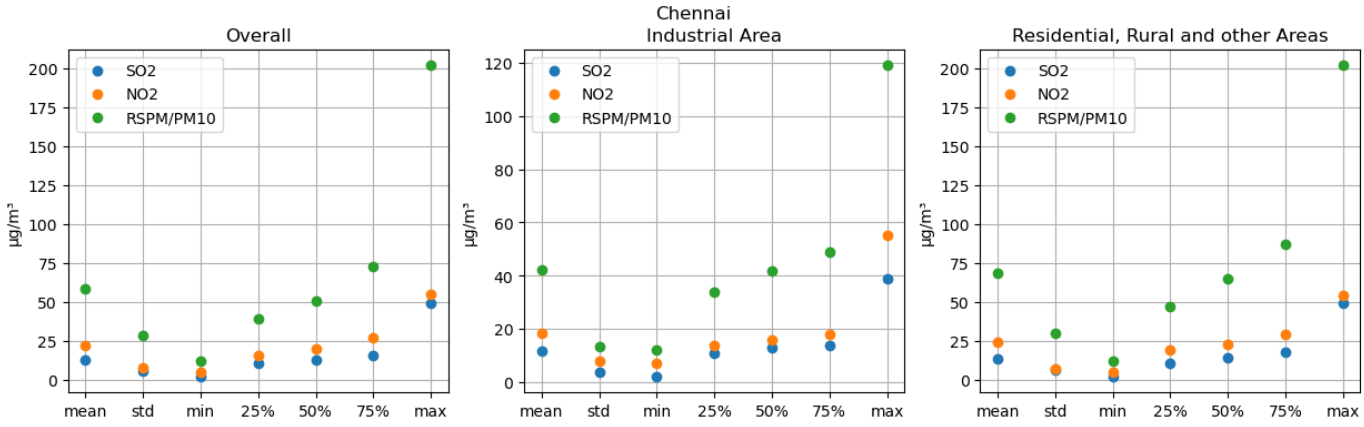
Statistics of SO2, NO2, and RSPM/PM10 levels across different monitoring stations, cities, or areas

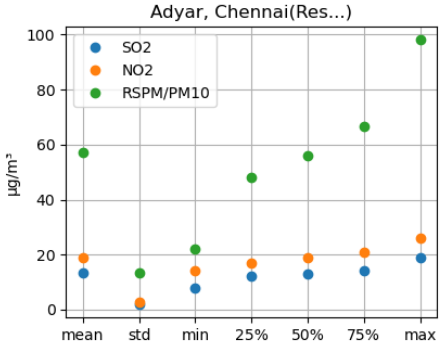
```
unique_stn = stn_df["name"].to_list()
computed_statistics = {}
for area_, data_ in data_by_area.items():
    computed_info = {
        "statistics": data_.describe(),
        "stat_by_loctype" : {},
        "stations": {

        }
    }
    for area_type in unique_loctyp:
        sub_group = data_[data_["loctype"] == area_type]
        if len(sub_group) > 0:
            computed_info["stat_by_loctype"][area_type] =
sub_group.describe()

    for station,sub_group in data_.groupby('loc'):
```

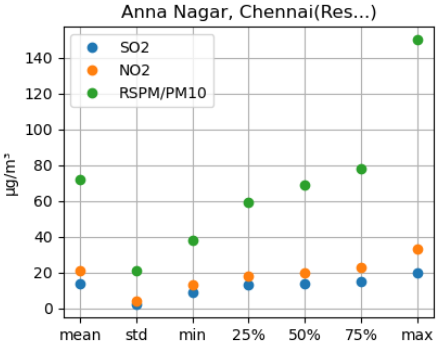
```
        sub_group = data_[data_["loc"] == station]

        computed_info["stations"][station] = {"area_type":
sub_group["loctype"].iloc[0], "statistics":sub_group.describe()}

    computed_statistics[area_] = computed_info
```

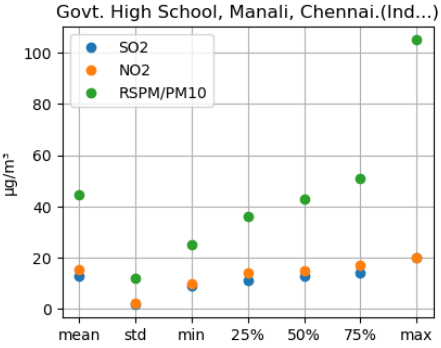
```
def draw_(stat,title,ax):
    stat.iloc[1:,:] .plot(kind="line",linestyle='',marker="o",ax=ax)
    ax.grid()
    ax.set_ylabel("µg/m³")
    ax.set_title(title)
def foo(area):
    info = computed_statistics[area]

    a_n = len(info["stat_by_loctype"])+1
    if area == "Salem":
        a_n +=1

    fig, axes = plt.subplots(nrows=1,ncols=a_n,figsize=(5*a_n,4))
    draw_(info["statistics"],"Overall",axes[0])
    i=1
    for area_type, stat in info["stat_by_loctype"].items():
        draw_(stat,area_type,axes[i])
        i+=1
    plt.suptitle(area)
    if area == "Salem":
        station = 'Sowdeswari College Building, Salem'
        draw_(info["stations"][station]["statistics"], station + f"
({info['stations'][station]['area_type'][:3]}...)", axes[i])
        plt.show()
        return
    plt.show()

    stations = info['stations'].keys()
    q, r = divmod(len(stations), 3)
    col_n =q + (1 if r > 0 else 0)
    fig, axes = plt.subplots(nrows=col_n, ncols=3, figsize=(5 * 3, 4 *
col_n))

    for i, station in enumerate(stations):
        row, col = divmod(i, 3)
```

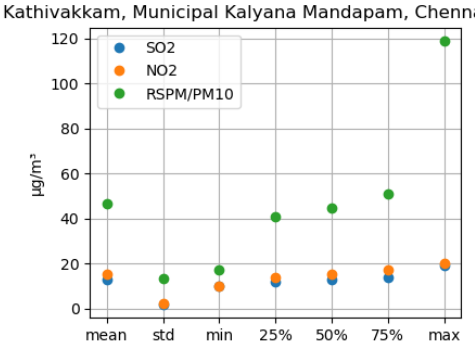
```
        ax = axes[row]
        if col_n >1:
            ax = axes[row][col]
        else:
            ax = axes[col]

        draw_(info["stations"][station]["statistics"], station + f"
({info['stations'][station]['area_type'][:3]}...)", ax)

    if r > 0:
        for j in range(r, 3):
            if col_n>1:
                fig.delaxes(axes[q, j])
            else:
                fig.delaxes(axes[j])
    plt.show()

for area in unique_areas:
    foo(area)
```

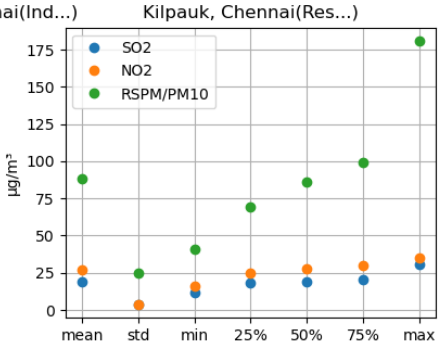


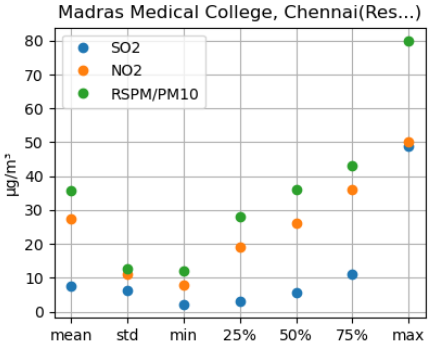

Adyar, Chennai(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

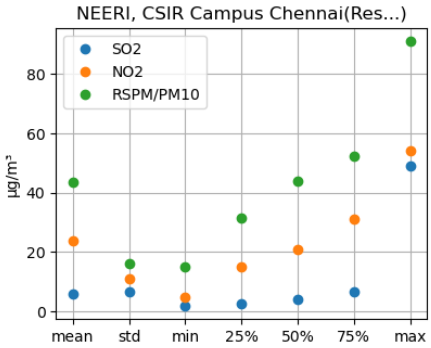

Anna Nagar, Chennai(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

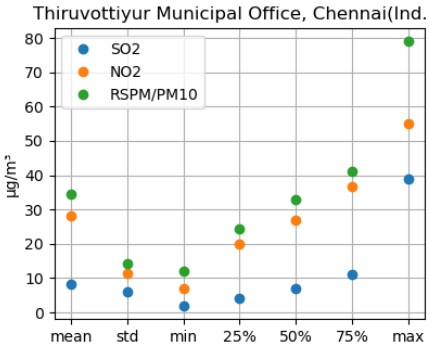

Govt. High School, Manali, Chennai.(Ind...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

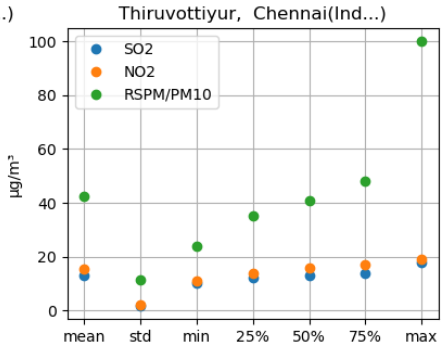

Kathivakkam, Municipal Kalyana Mandapam, Chennai(Ind...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

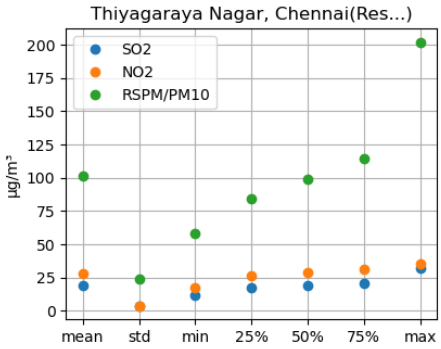

Kilpauk, Chennai(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

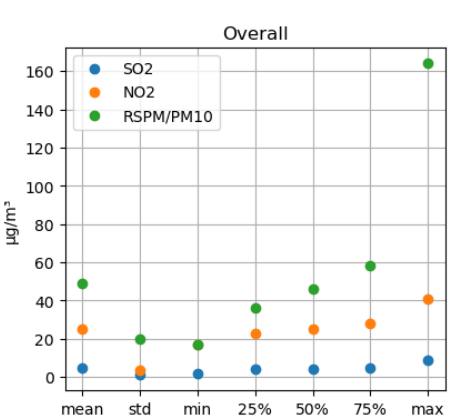

Madras Medical College, Chennai(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

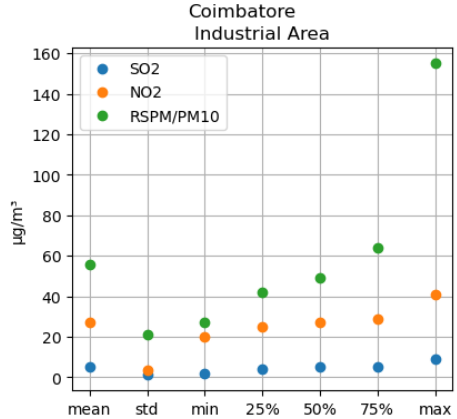

NEERI, CSIR Campus Chennai(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

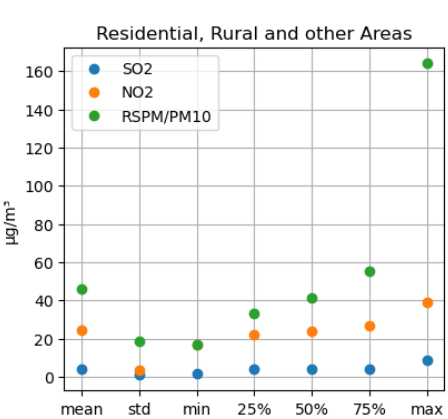

Thiruvottiyur Municipal Office, Chennai(Ind...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

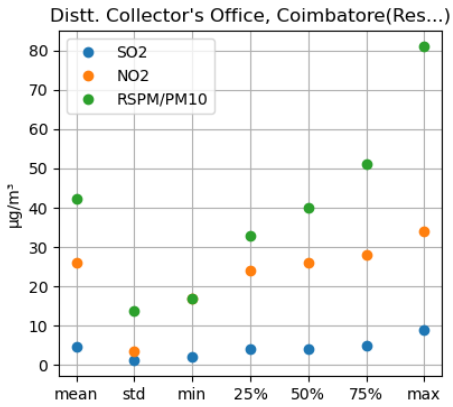

Thiruvottiyur, Chennai(Ind...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

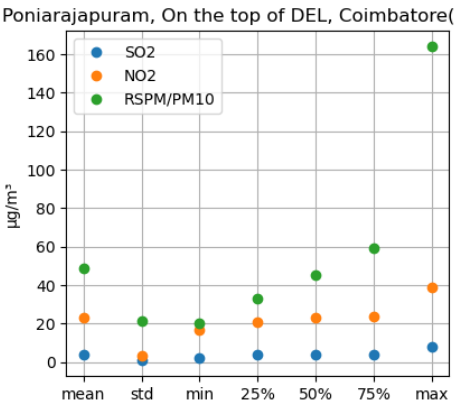

Thiyagaraya Nagar, Chennai(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

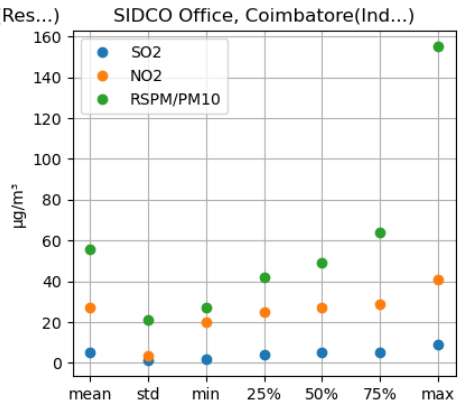

Overall
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

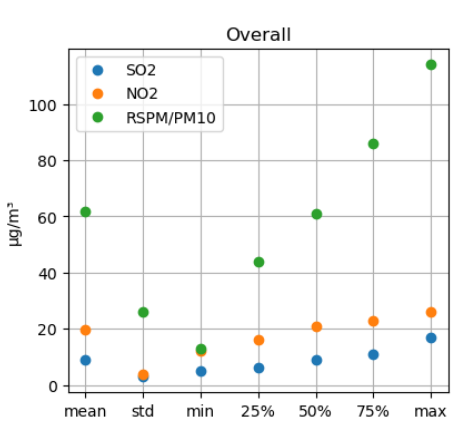

Coimbatore
Industrial Area
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

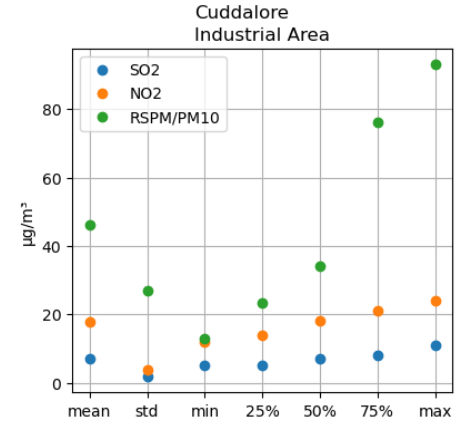

Residential, Rural and other Areas
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

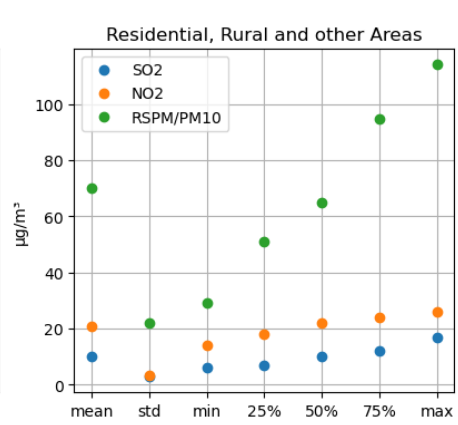

Distt. Collector's Office, Coimbatore(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

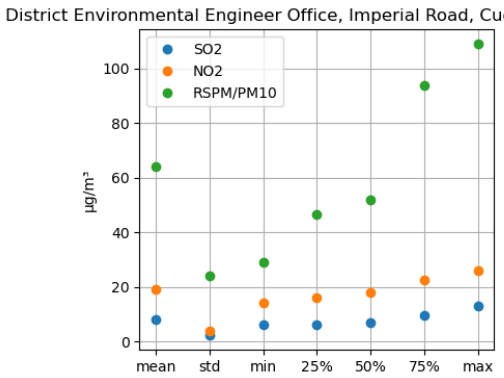

Poniarajapuram, On the top of DEL, Coimbatore(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

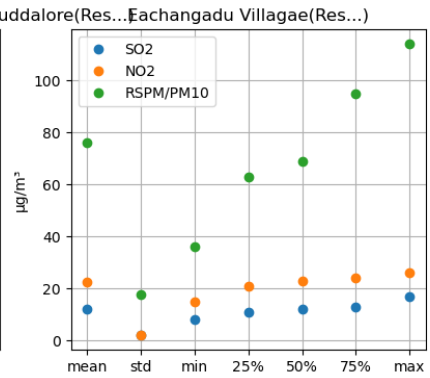

SIDCO Office, Coimbatore(Ind...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

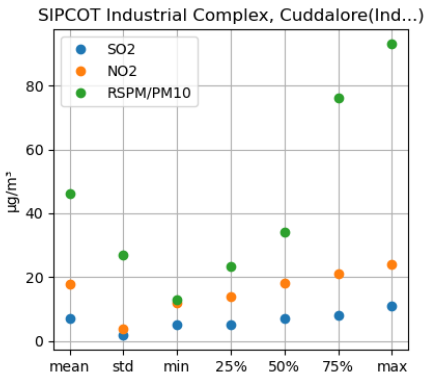

Overall
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

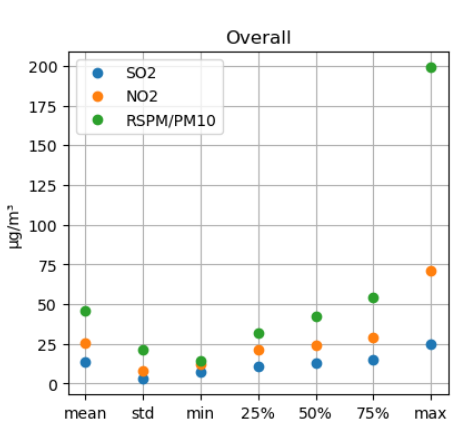

Cuddalore
Industrial Area
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

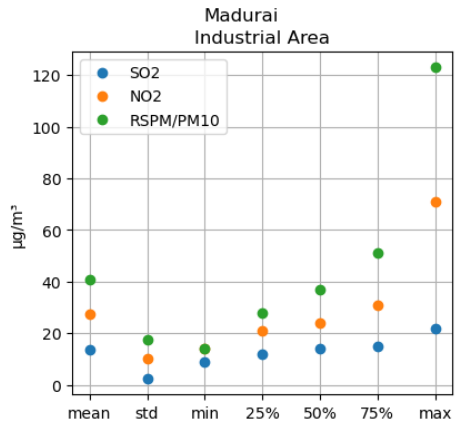

Residential, Rural and other Areas
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

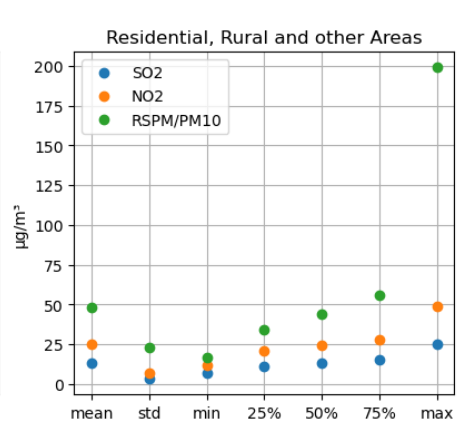

District Environmental Engineer Office, Imperial Road, Cuddalore(Res...
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max


Eachangadu Villagae(Res...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

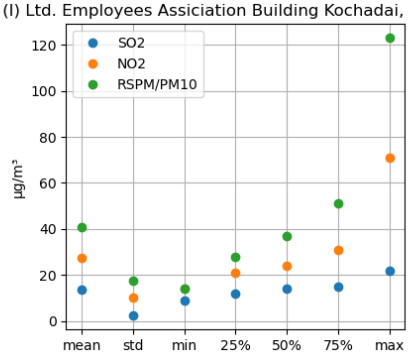

SIPCOT Industrial Complex, Cuddalore(Ind...)
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max


Overall
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

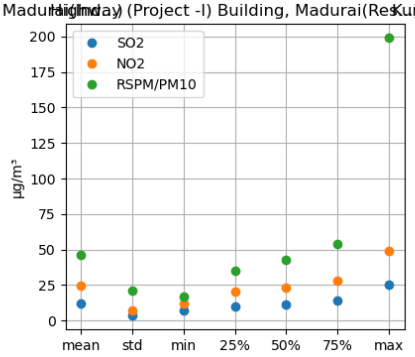

Madurai
Industrial Area
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max


Residential, Rural and other Areas
SO2
NO2
RSPM/PM10
µg/m³
mean
std
min
25%
50%
75%
max

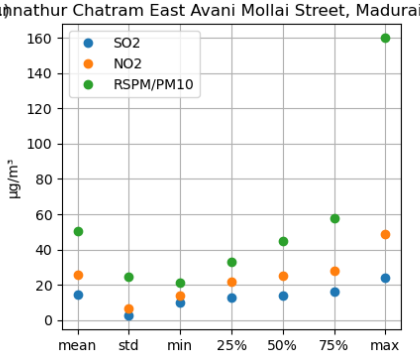
Fenner (I) Ltd. Employees Assiciation Building Kochadai, Madurai(Res...)



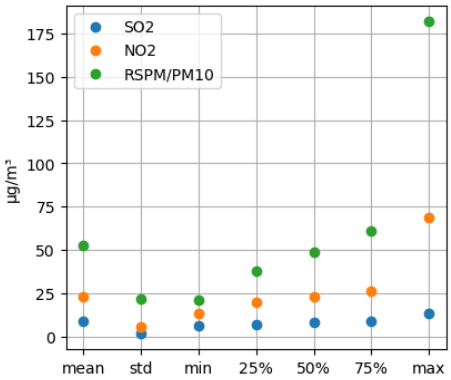
Highway (Project -I) Building, Madurai(Res...)



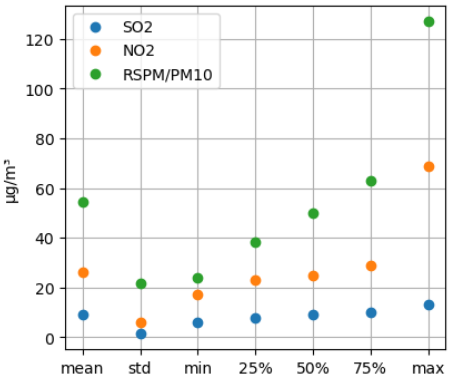
Kunnathur Chatram East Avani Mollai Street, Madurai(Res...)



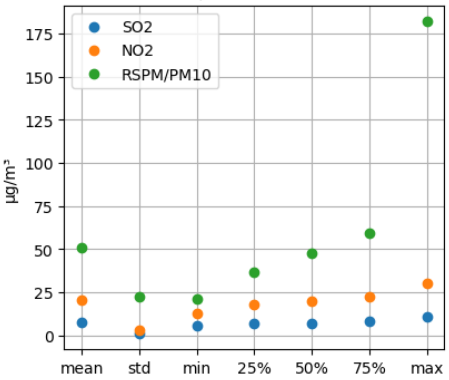
Overall



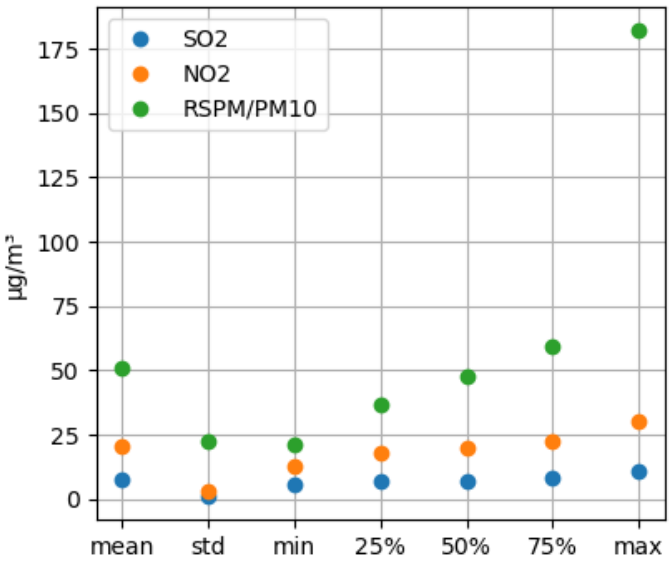
Mettur
Industrial Area



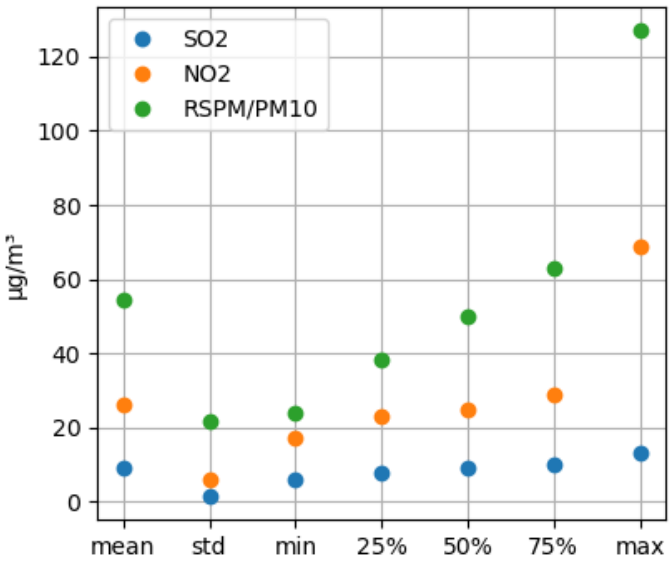
Residential, Rural and other Areas



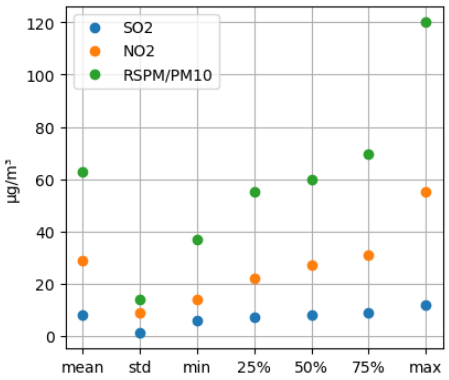
Raman Nagar, Mettur(Res...)



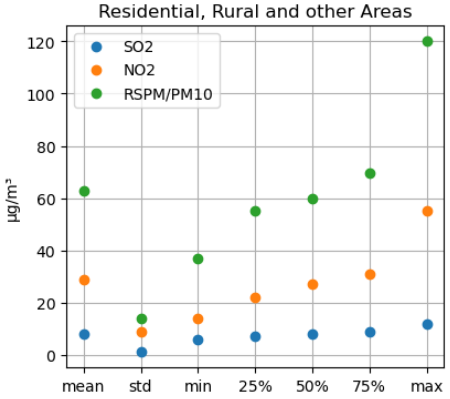
SIDCO Industrial Complex, Mettur(Ind...)



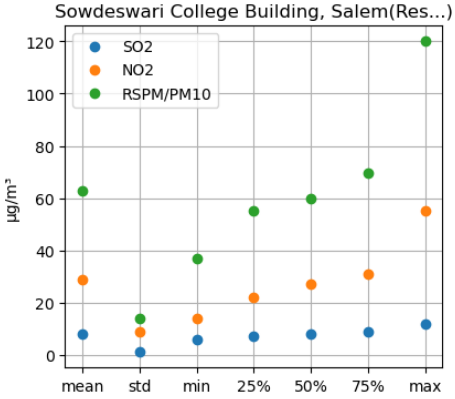
Overall



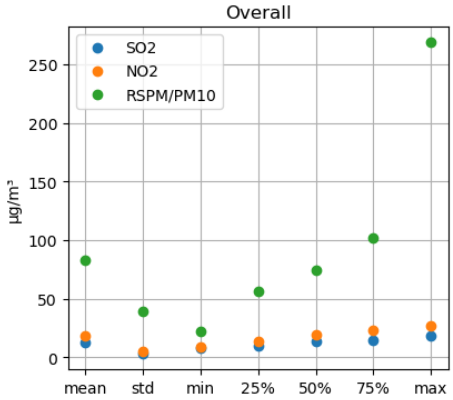
Salem
Residential, Rural and other Areas

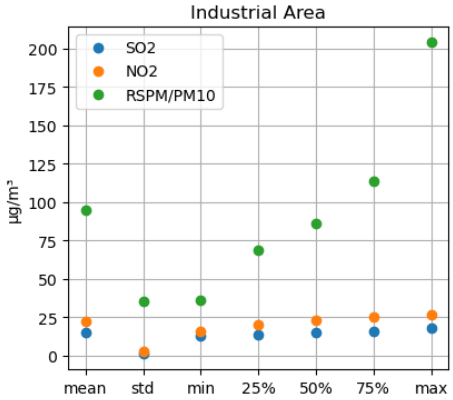


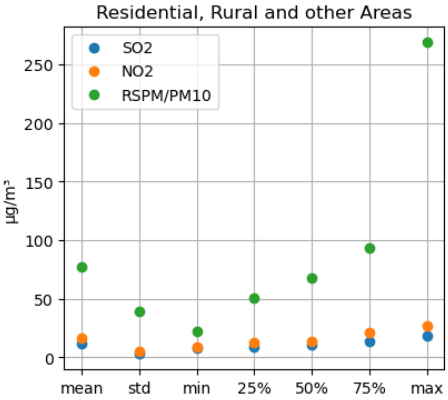
Sowdeswari College Building, Salem(Res...)

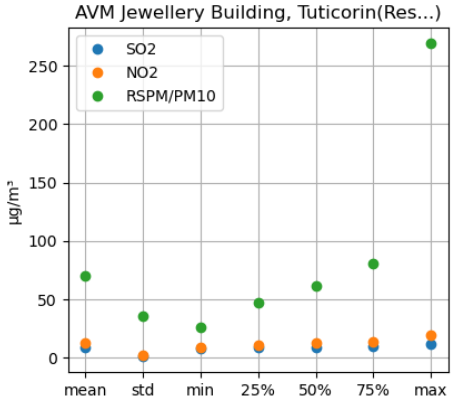


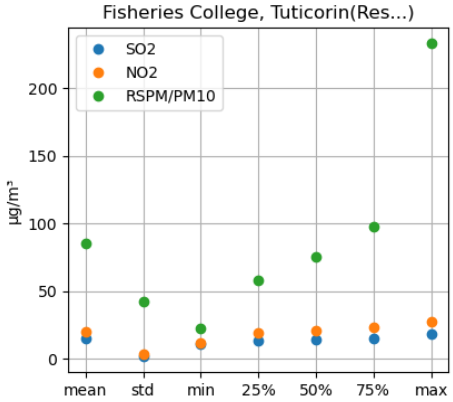
Thoothukudi

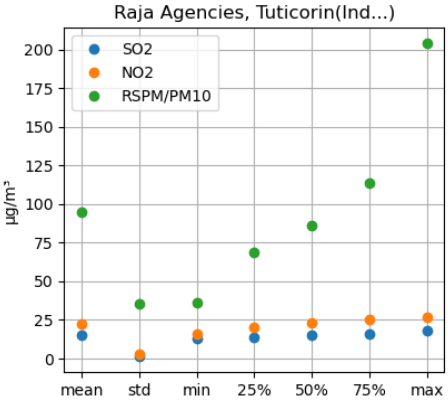




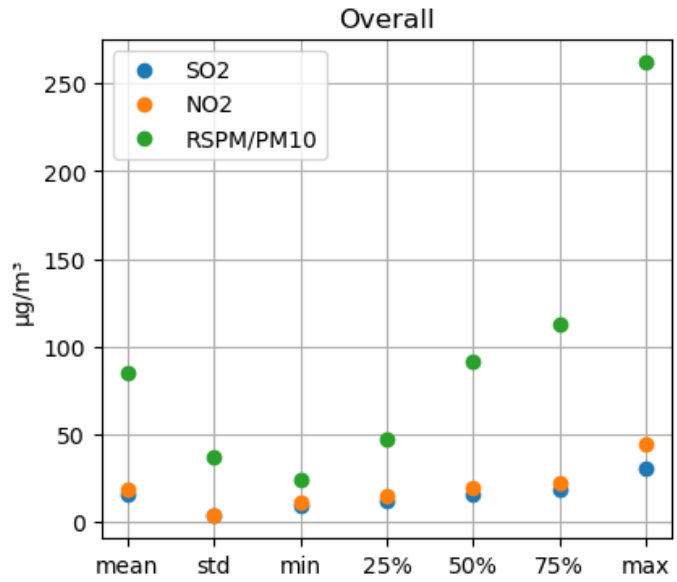


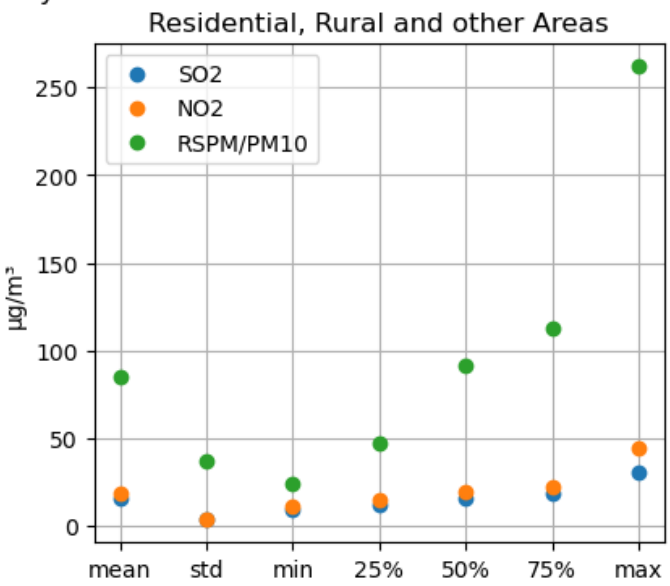


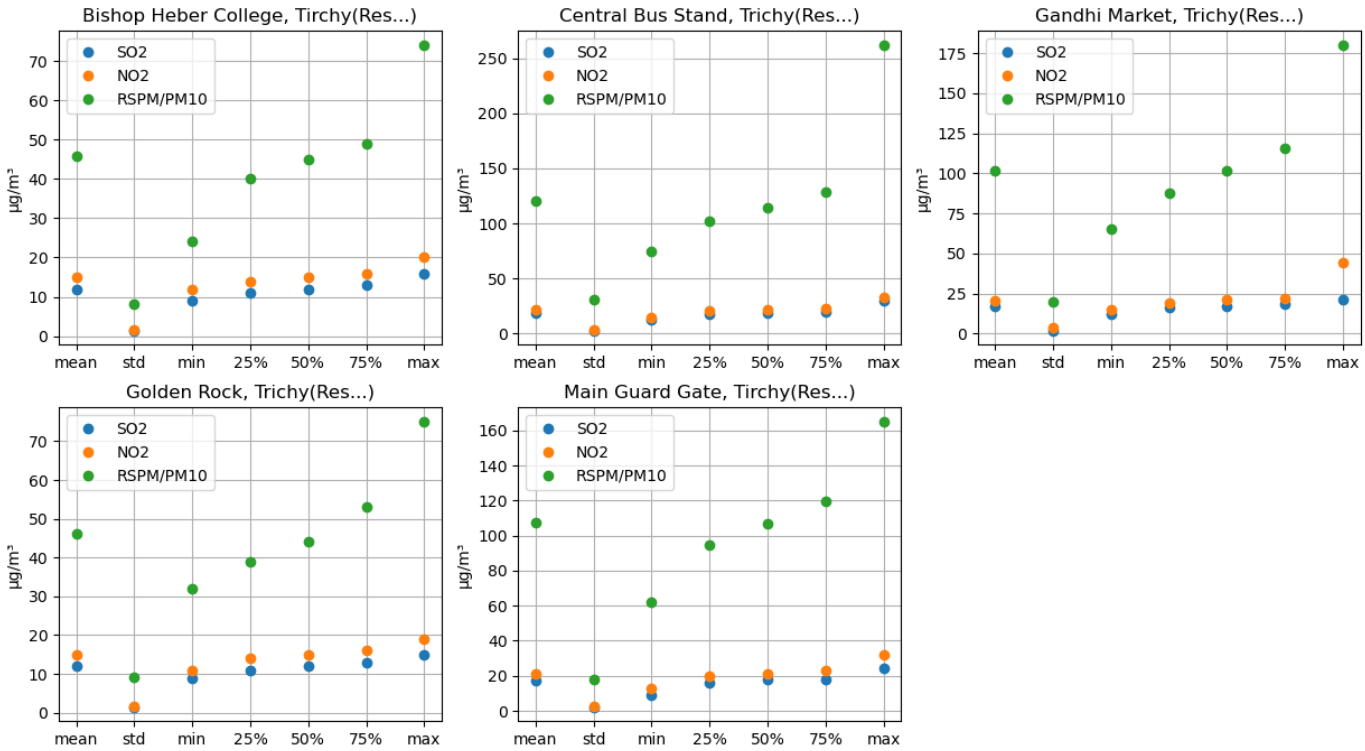




Trichy







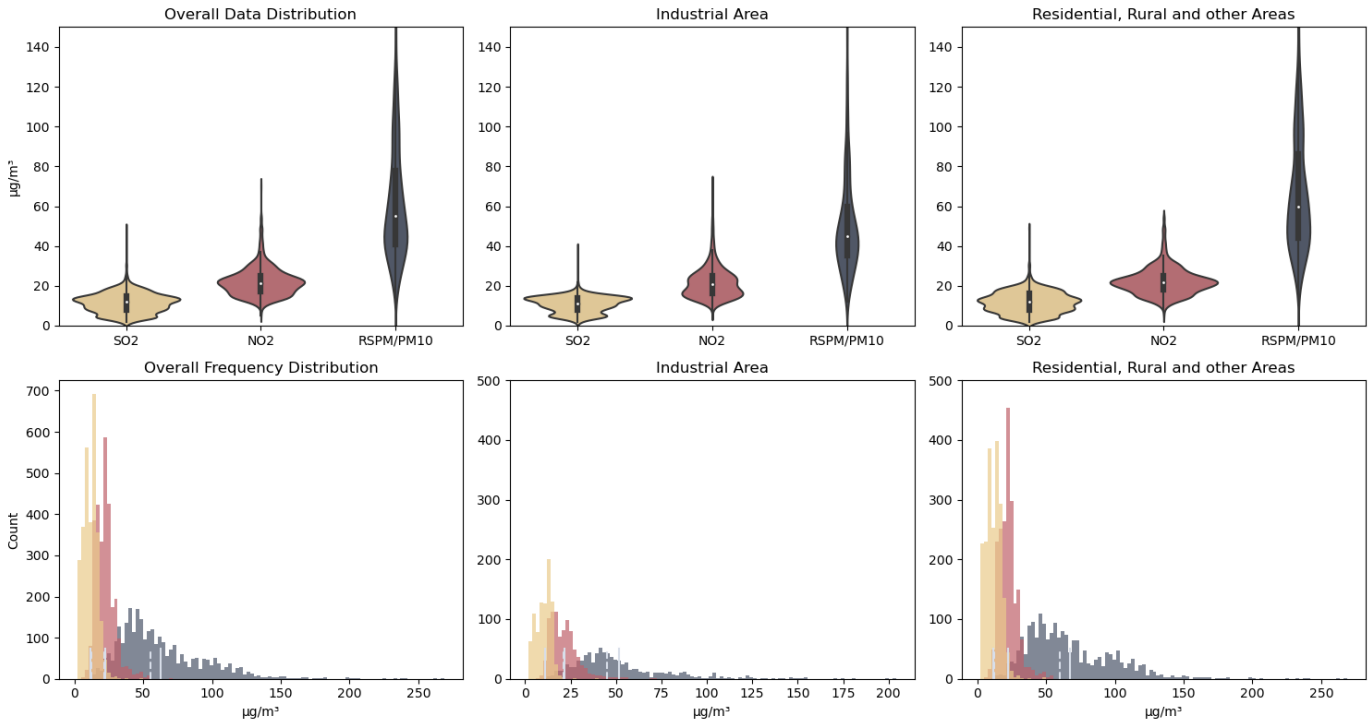
Overall Frequency & Data Distribution of SO2, NO2 & RSPM/PM10

```
desc = data.describe()
desc
```

	SO2	NO2	RSPM/PM10
count	2862.000000	2862.000000	2862.000000
mean	11.506988	22.135220	62.437456
std	5.050855	7.133291	31.277419
min	2.000000	5.000000	12.000000
25%	8.000000	17.000000	41.000000
50%	12.000000	21.500000	55.000000
75%	15.000000	25.000000	78.000000
max	49.000000	71.000000	269.000000

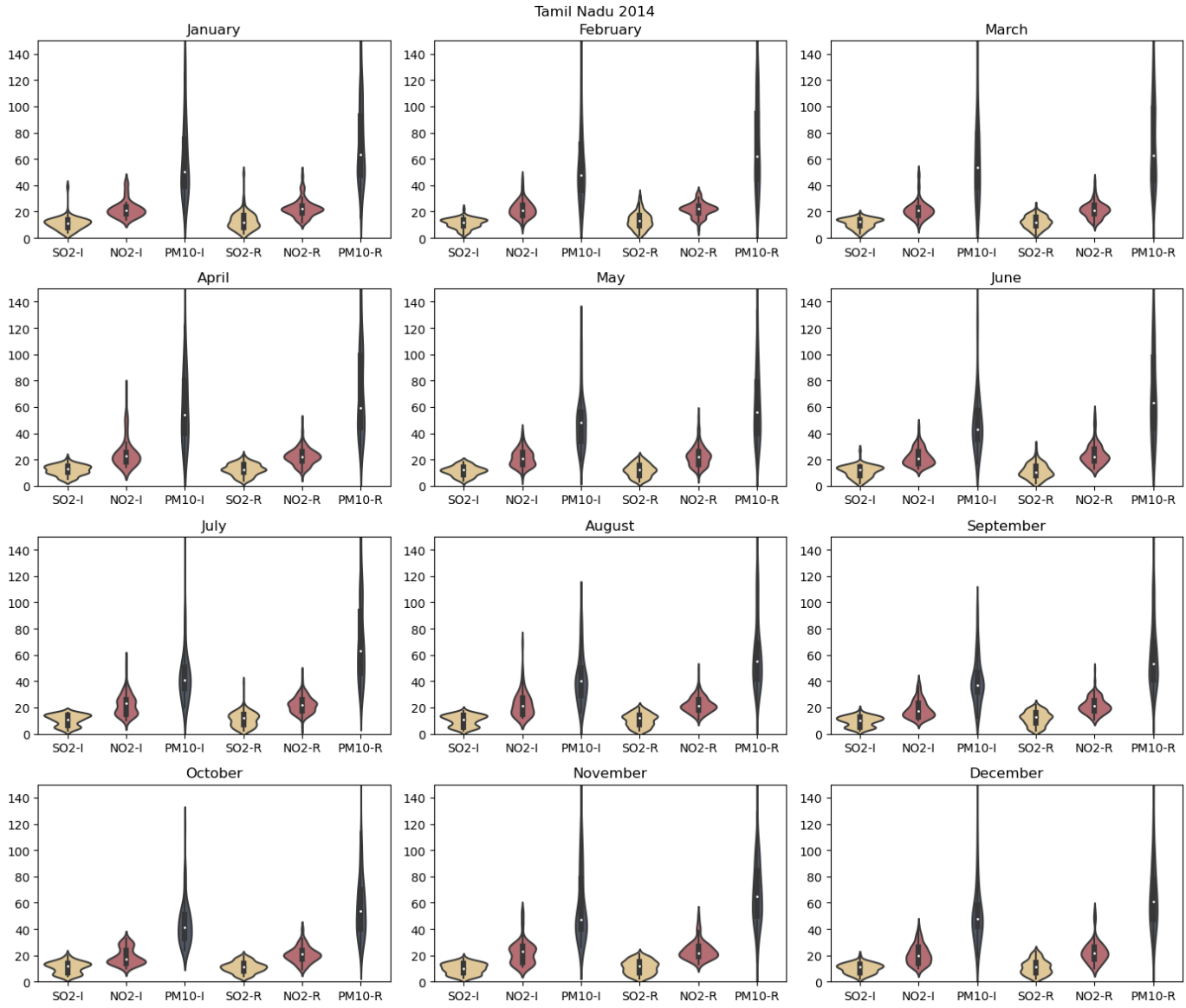
```
fig, axes = plt.subplots(nrows=2,ncols=3,figsize=(15,8))
sns.violinplot(data,palette=[so2_color,no2_color,pm10_color],ax=axes[0][0])
axes[0][0].set_ylabel("µg/m³")
axes[0][0].set_title("Overall Data Distribution")
axes[0][0].set_ylim(0,150)
for i in range(2):
    ax = axes[0][i+1]
    sns.violinplot(data[data["loctype"]==unique_loctyp[i]],palette=
```

```
[so2_color,no2_color,pm10_color],ax=ax)
    ax.set_title(unique_loctyp[i])
    ax.set_ylim(0,150)


sns.histplot(data,bins=100,palette=
[so2_color,no2_color,pm10_color],alpha=0.7, linewidth=0,ax=axes[1]
[0],legend=False)

for i in range(3):
    axes[1][0].axvline(desc.loc["mean"][i], color="#D8DEE9", linestyle='-',
label='Mean',ymax=0.1)
    axes[1][0].axvline(desc.loc["50%"][i], color="#D8DEE9", linestyle='--',
label='50%',ymax=0.1)

axes[1][0].set_ylabel("Count")
axes[1][0].set_xlabel("µg/m³")
axes[1][0].set_title("Overall Frequency Distribution")
for i in range(2):
    data_ = data[data["loctype"]==unique_loctyp[i]]
    ax = axes[1][i+1]
    sns.histplot(data_,bins=100,palette=
[so2_color,no2_color,pm10_color],alpha=0.7, linewidth=0,ax=ax,legend=False)
    mean_values = data_.mean(numeric_only=True)
    median_values = data_.median(numeric_only=True)
    for j in range(3):
        ax.axvline(mean_values[j], color="#D8DEE9", linestyle='-',
label='Mean',ymax=0.1)
        ax.axvline(median_values[j], color="#D8DEE9", linestyle='--',
label='50%',ymax=0.1)
    ax.set_ylabel("")
    ax.set_xlabel("µg/m³")
    ax.set_ylim(0,500)
    ax.set_title(unique_loctyp[i])
plt.tight_layout()
plt.show()
```



Overall Data Distribution by Month

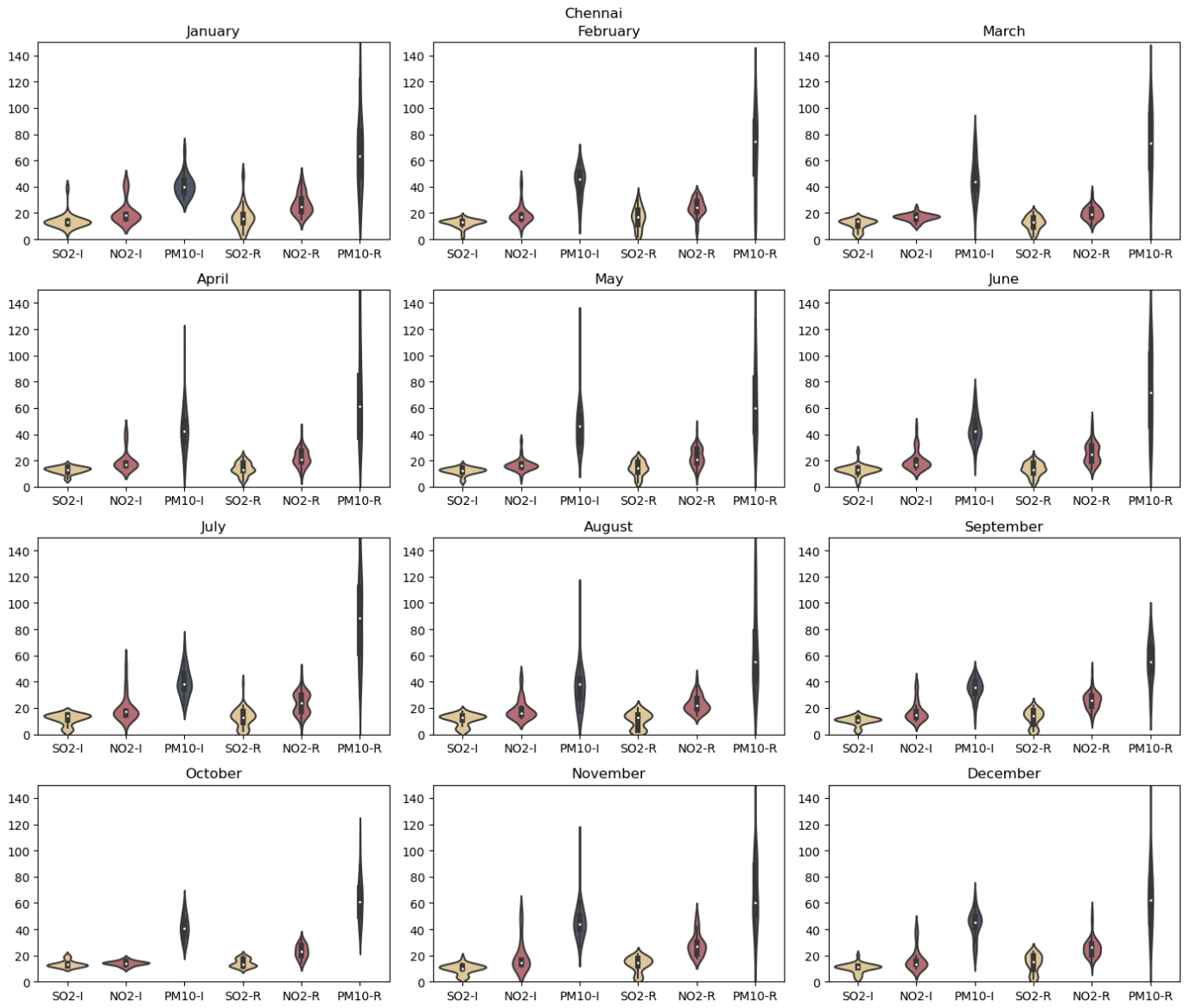
```
def make_temp_loctyp(data):
    tempdf = pd.DataFrame()
    for n, x in data.groupby("loctype"):
        tempdfn = pd.DataFrame()
        tempdfn["SO2-"+n[0]] = x["SO2"]
        tempdfn["NO2-"+n[0]] = x["NO2"]
        tempdfn["PM10-"+n[0]] = x["RSPM/PM10"]
        tempdf = pd.concat([tempdf, tempdfn], ignore_index=True)

    return tempdf

def draw_():
    fig,axes = plt.subplots(nrows=4,ncols=3,figsize=(14,12))
    for i,(group, data_) in
enumerate(data.groupby(data["date"].dt.to_period('M'))):
        tempdf = make_temp_loctyp(data_)
        ax = axes[i//3][i%3]
        sns.violinplot(tempdf,ax=ax,palette=
[so2_color,no2_color,pm10_color,so2_color,no2_color,pm10_color])
        ax.set_title(group.strftime("%B"))
        ax.set_ylim(0,150)
    plt.suptitle("Tamil Nadu 2014")
    plt.tight_layout()
    plt.show()
draw_()
```

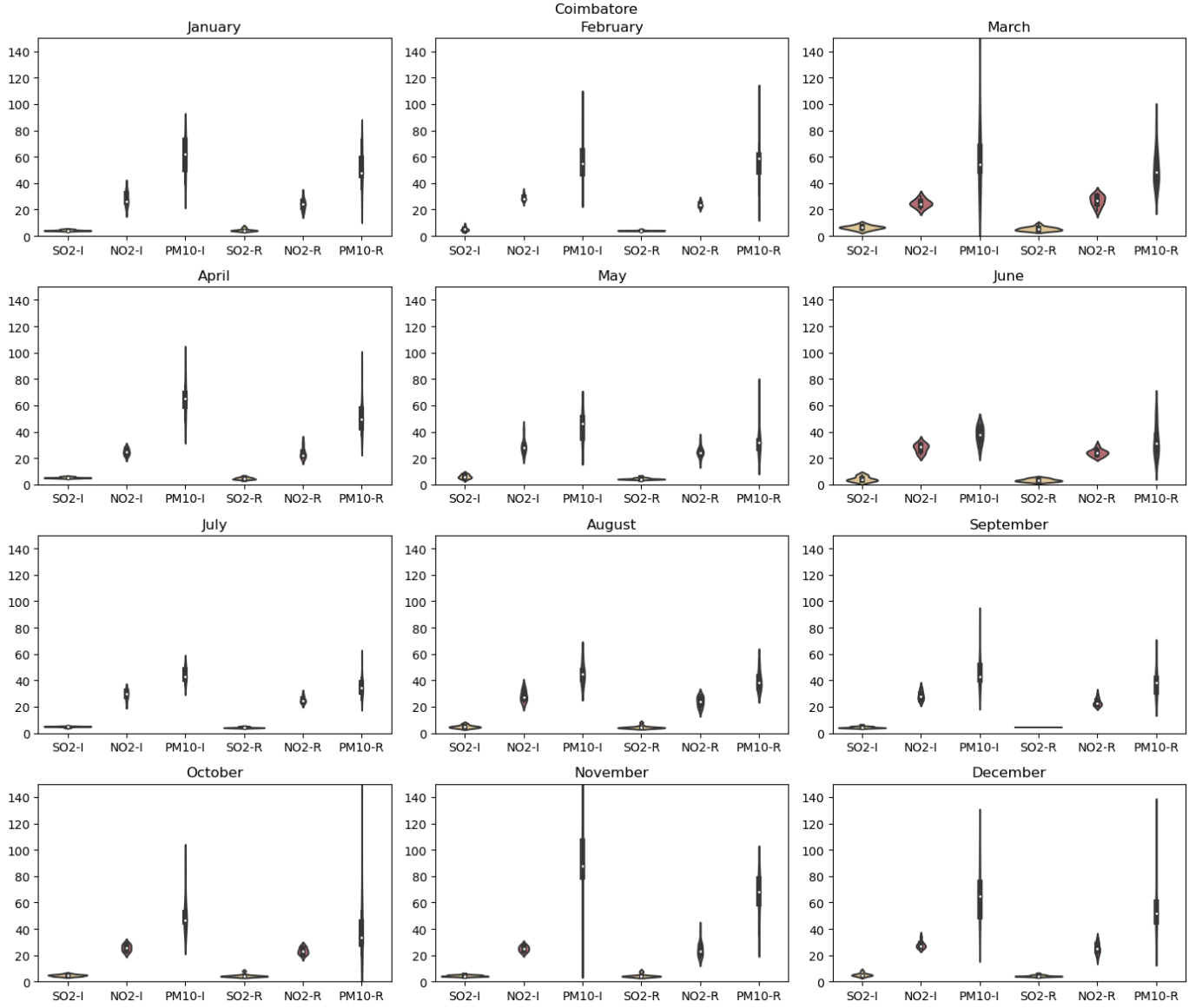


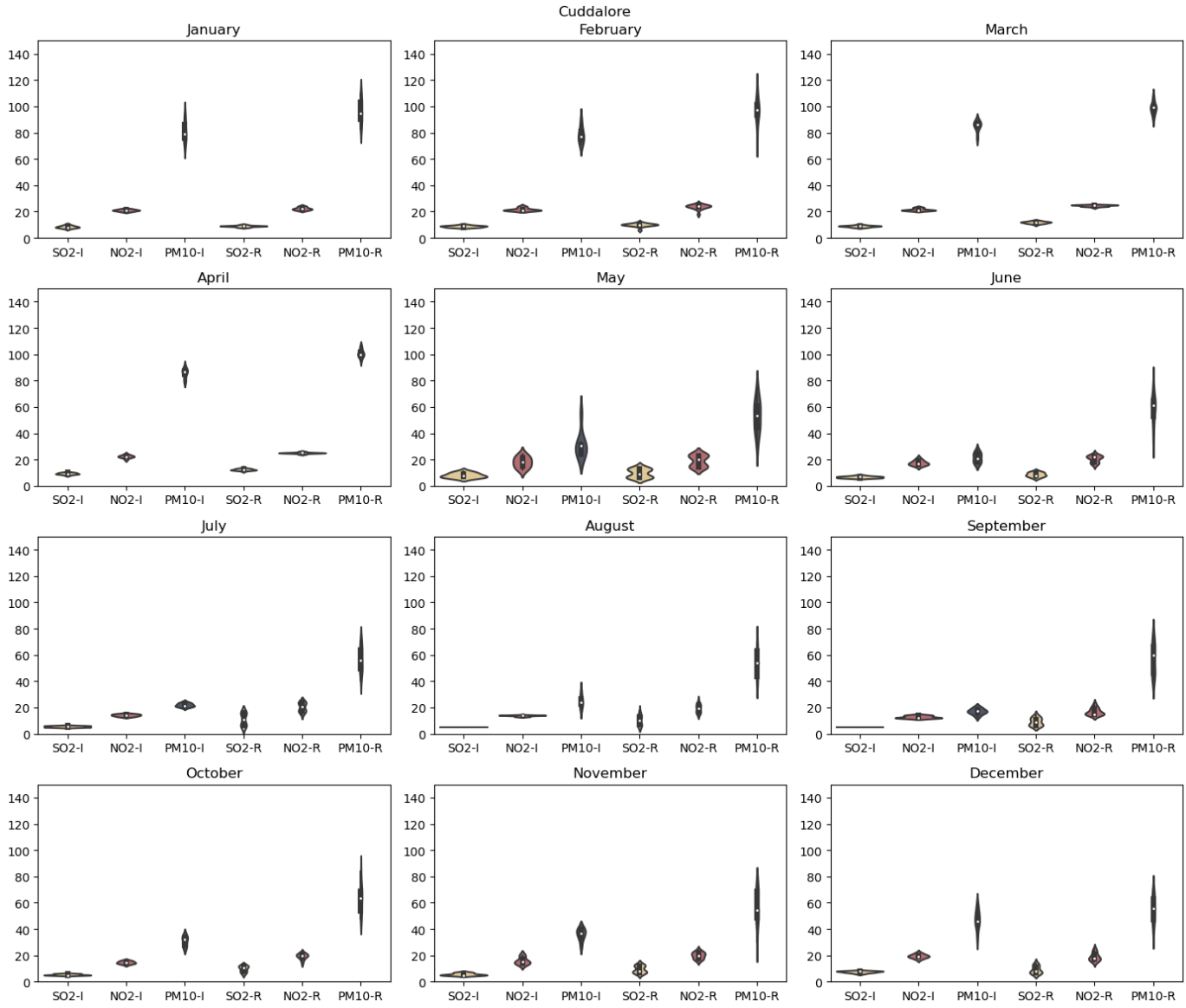
Data Distribution of Every Area and location type by Month

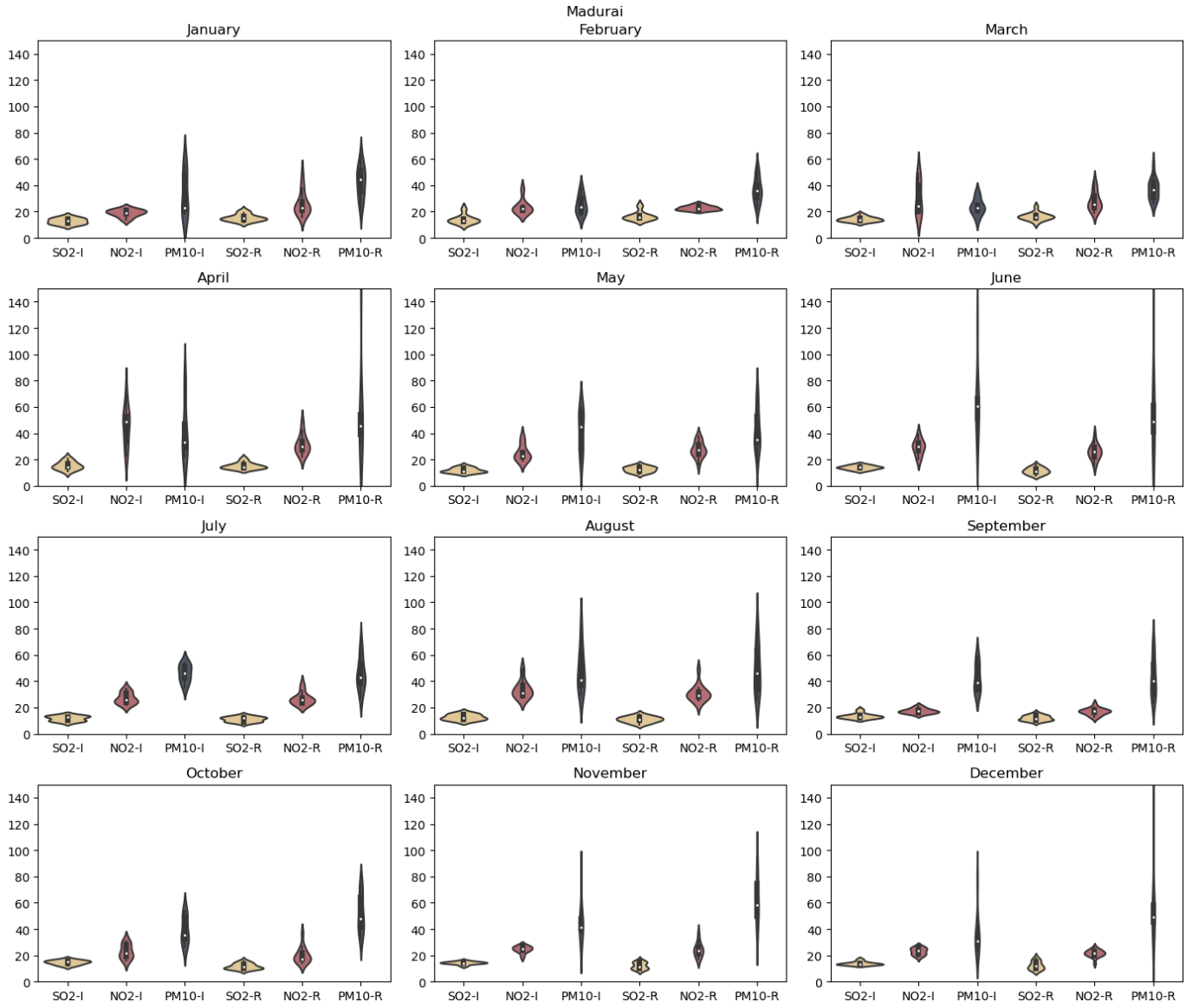
```
def draw_(area):
    fig,axes = plt.subplots(nrows=4,ncols=3,figsize=(14,12))
    for i,(group, data_) in
enumerate(data_by_area[area].groupby(data_by_area[area]
["date"].dt.to_period('M'))):

        tempdf = make_temp_loctyp(data_)
        ax = axes[i//3][i%3]
        sns.violinplot(tempdf,ax=ax,palette=
[so2_color,no2_color,pm10_color,so2_color,no2_color,pm10_color])
        ax.set_title(group.strftime("%B"))
        ax.set_ylim(0,150)
    plt.suptitle(area)
    plt.tight_layout()
    plt.show()
```

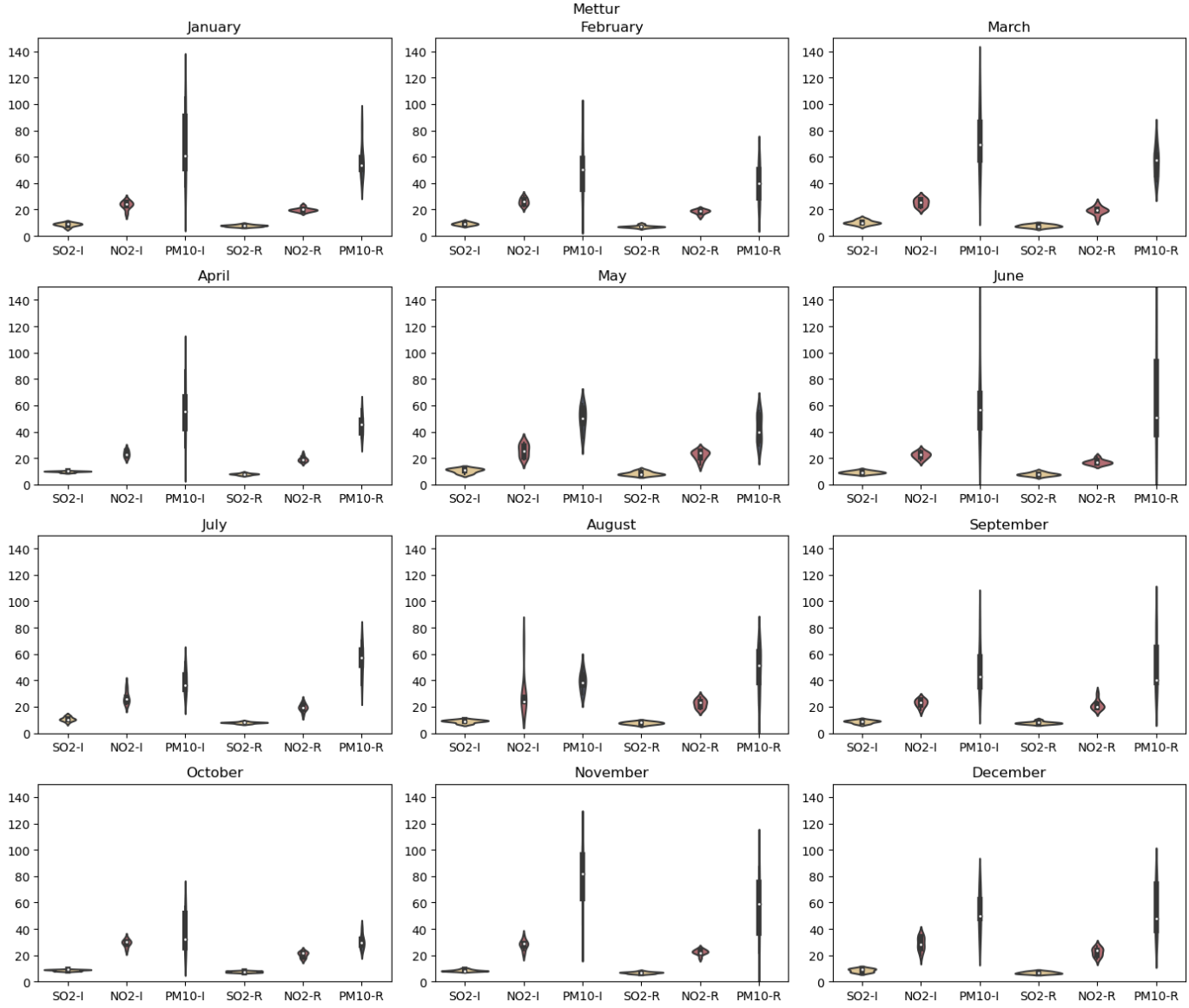
```
for area in unique_areas:
    draw_(area)
```

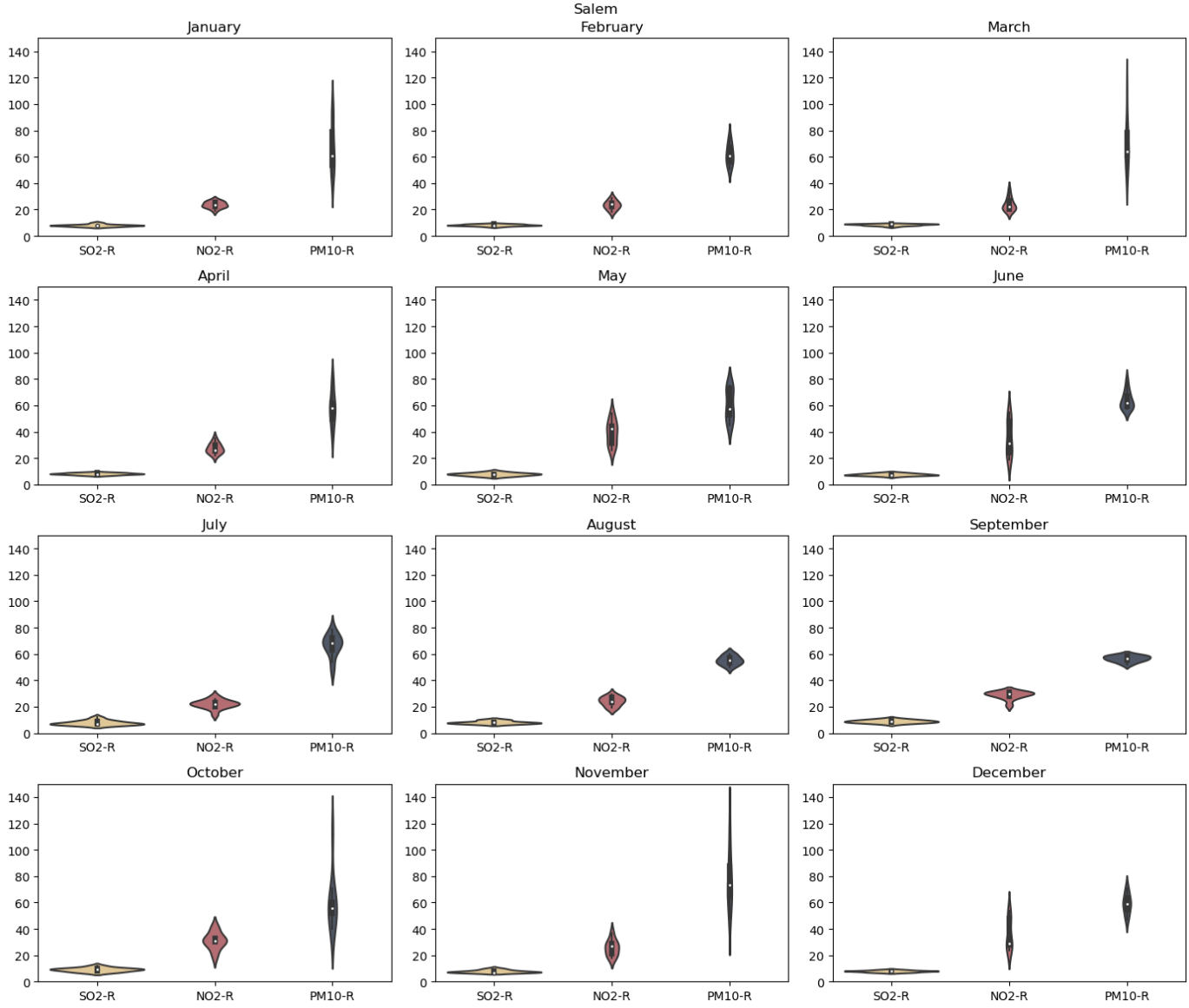


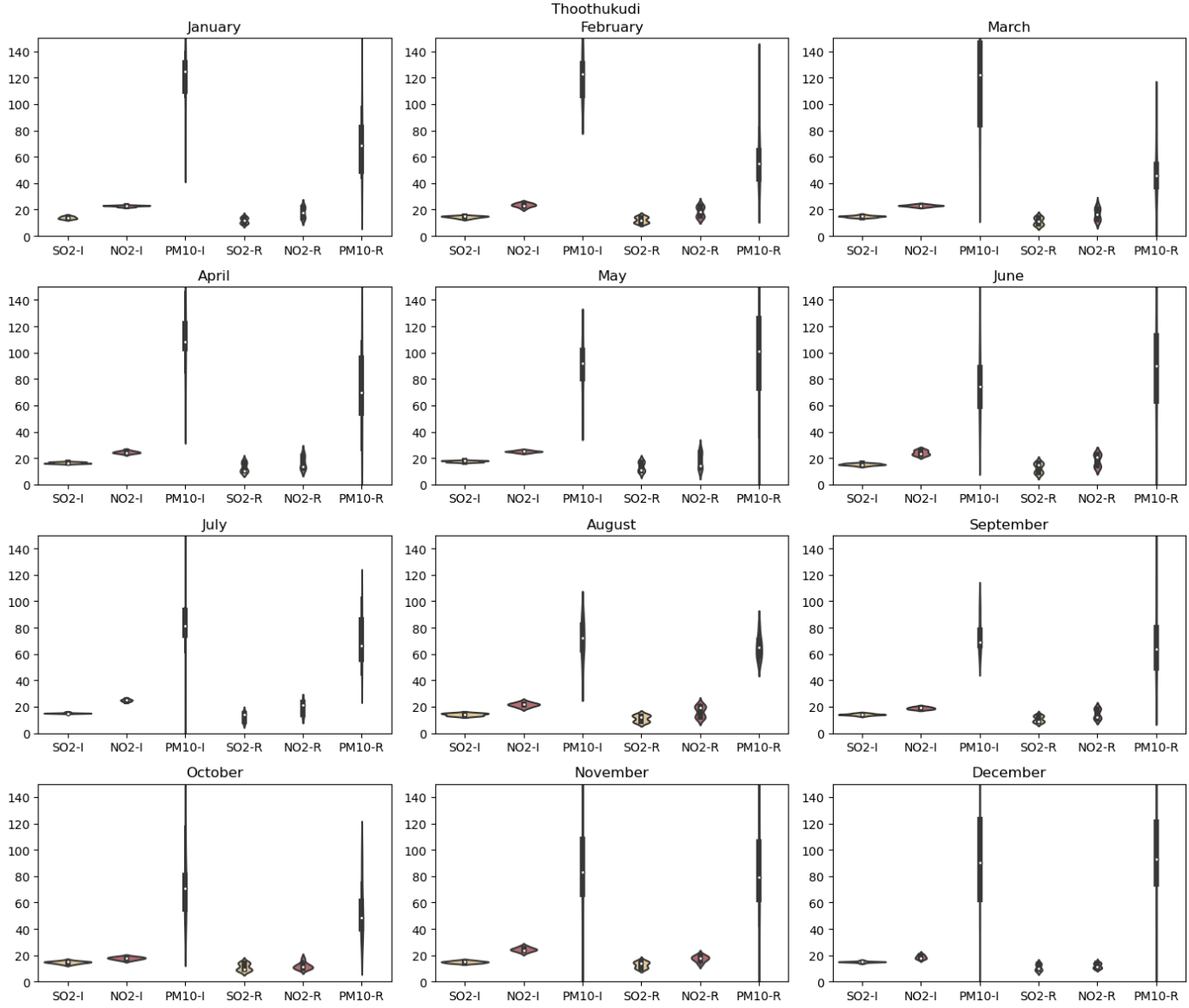

Coimbatore
January
February
March
April
May
June
July
August
September
October
November
December
SO2-I
NO2-I
PM10-I
SO2-R
NO2-R
PM10-R
0
20
40
60
80
100
120
140

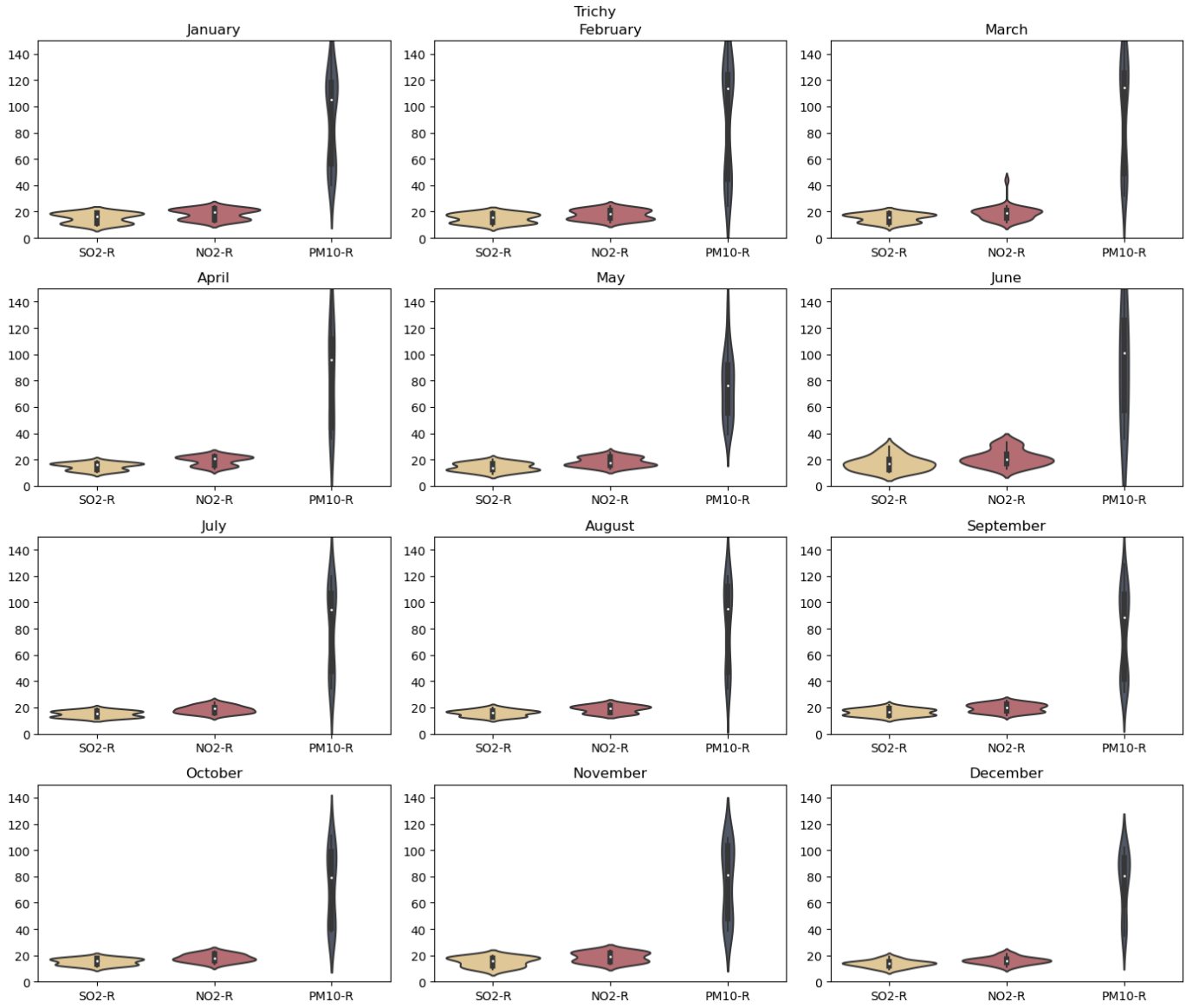

Cuddalore
January
February
March
April
May
June
July
August
September
October
November
December
SO2-I
NO2-I
PM10-I
SO2-R
NO2-R
PM10-R
0
20
40
60
80
100
120
140


Madurai
January
February
March
April
May
June
July
August
September
October
November
December
0
20
40
60
80
100
120
140
SO2-I
NO2-I
PM10-I
SO2-R
NO2-R
PM10-R


Mettur
January
February
March
April
May
June
July
August
September
October
November
December
SO2-I
NO2-I
PM10-I
SO2-R
NO2-R
PM10-R
0
20
40
60
80
100
120
140


Salem
January
February
March
April
May
June
July
August
September
October
November
December
SO2-R
NO2-R
PM10-R
0
20
40
60
80
100
120
140


Thoothukudi
January
February
March
April
May
June
July
August
September
October
November
December
0
20
40
60
80
100
120
140
SO2-I
NO2-I
PM10-I
SO2-R
NO2-R
PM10-R



Data Distribution of Every Area and location type by Weekdays (MON to SAT)

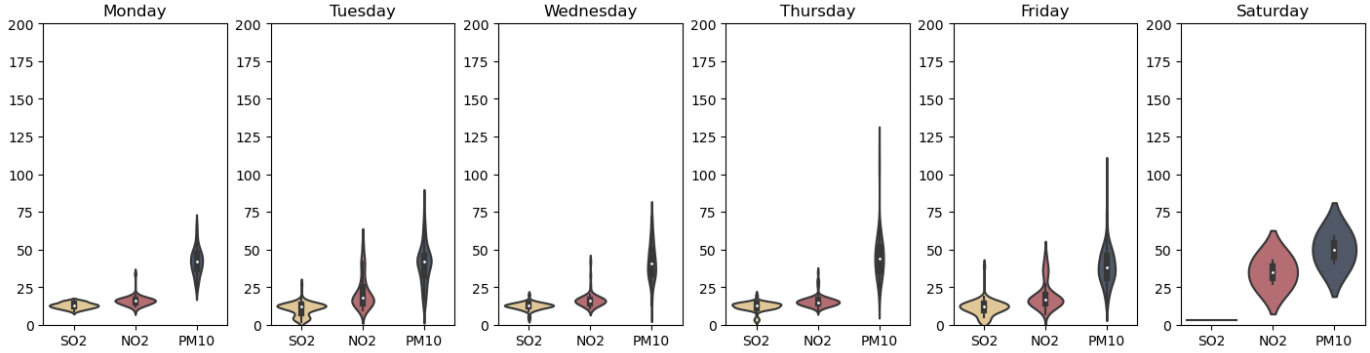
NOTE: NO DATA AVAILABLE FOR SUNDAYS, SO I REMOVED SUNDAY FROM THE LIST

```
# NO DATA AVAILABLE FOR SUNDAYS, SO I REMOVED SUNDAY FROM THE LIST
def draw_(area):
    day_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday']
    gb_loc = data_by_area[area].groupby("loctype")
    N_ = len(gb_loc)

    fig = plt.figure(constrained_layout=True,figsize=(14,4*N_))
    fig.suptitle(area)

    i = 0

    subfigs = fig.subfigures(nrows=N_, ncols=1,)
```

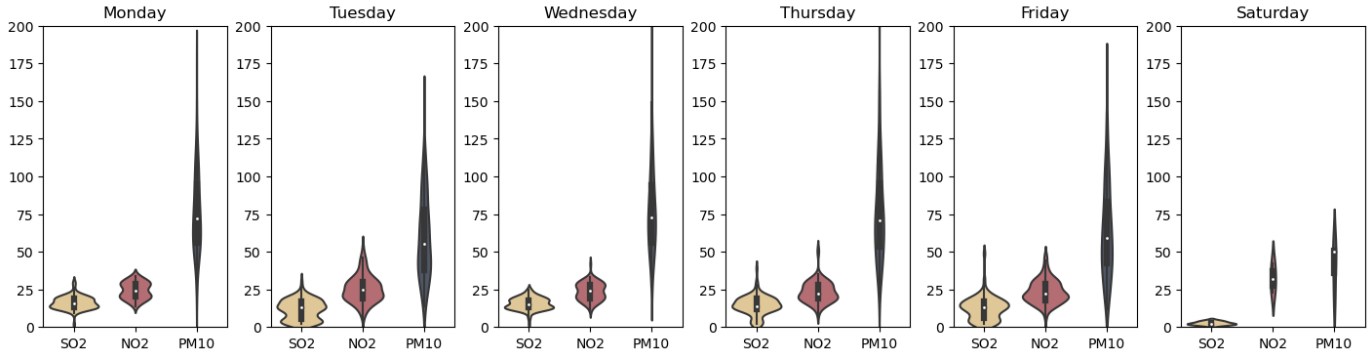
```
    for name, data_ in gb_loc:
        if N_ >1:
            subfig = subfigs[i]
        else: subfig = subfigs
        axs = subfig.subplots(nrows=1,ncols=len(day_names))

        subfig.suptitle(name,fontsize=12)

        by_week_days = defaultdict(list)

        for _,row in  data_.iterrows():

            by_week_days[row["date"].strftime("%A")].append(
                (row["SO2"], row["NO2"],row["RSPM/PM10"])
            )

        for idx, day in enumerate(day_names):
            values = np.array(by_week_days[day])
            if len(values)>0:
                temp = pd.DataFrame(values)
                temp.columns = ["SO2","NO2","PM10"]

                sns.violinplot(data=temp, ax=axs[idx],palette=
[so2_color,no2_color,pm10_color])

                axs[idx].set_ylim(0,200)
            else:
                axs[idx].axis('off')
                axs[idx].text(0.5, 0.5, "NA", fontsize=24, ha='center',
va='center')
            axs[idx].set_title(day)

        i+=1

    plt.show()


for area in data_by_area.keys():
    draw_(area)
```

Chennai

Industrial Area



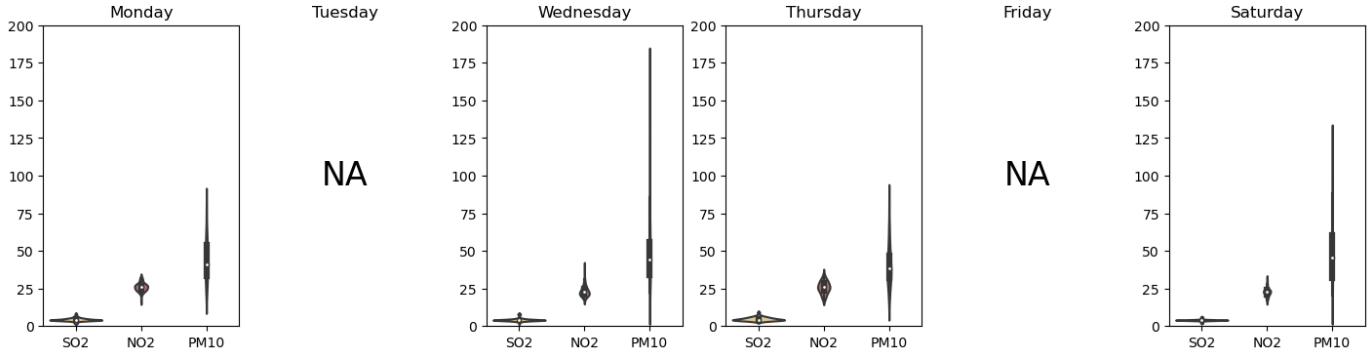
Residential, Rural and other Areas



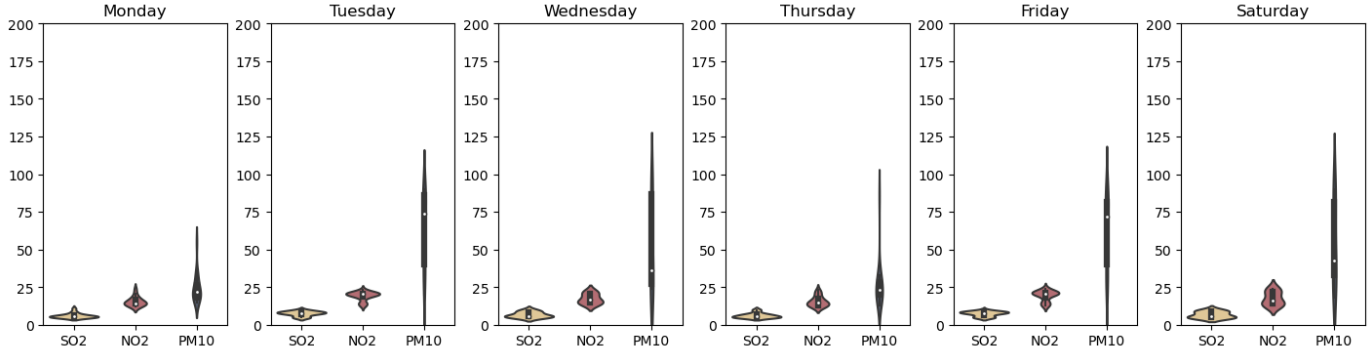
Coimbatore

Industrial Area

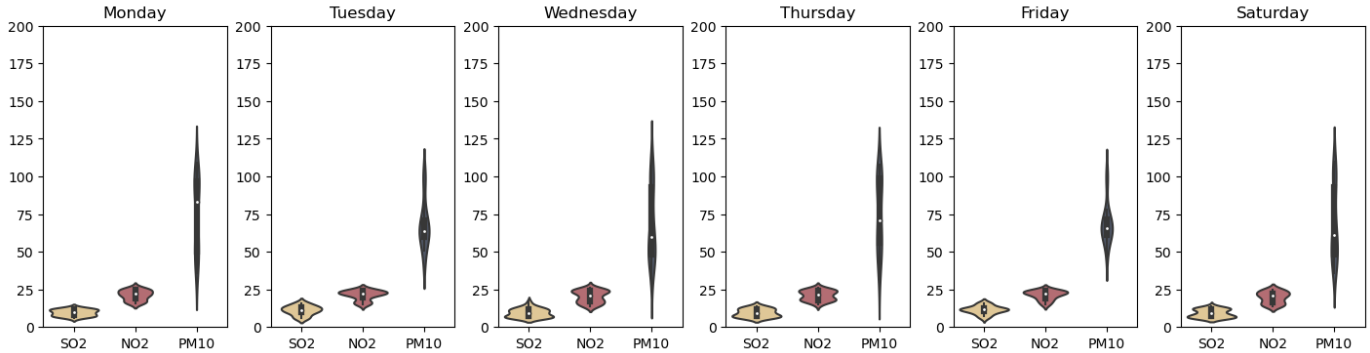
Residential, Rural and other Areas



Cuddalore
Industrial Area



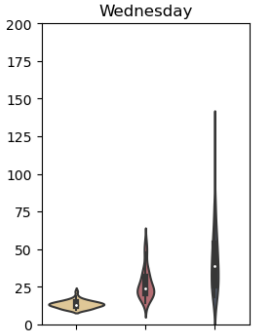
Residential, Rural and other Areas



Madurai
Industrial Area

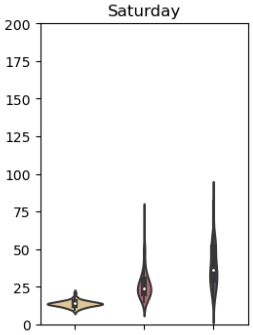
Monday NA

Tuesday NA

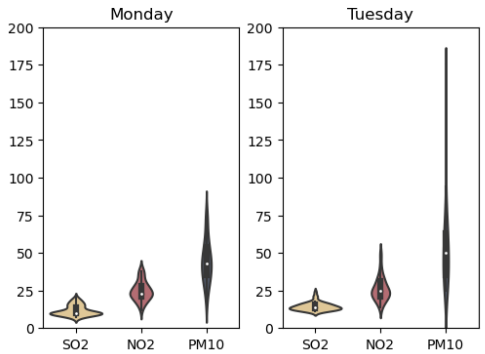


Thursday NA

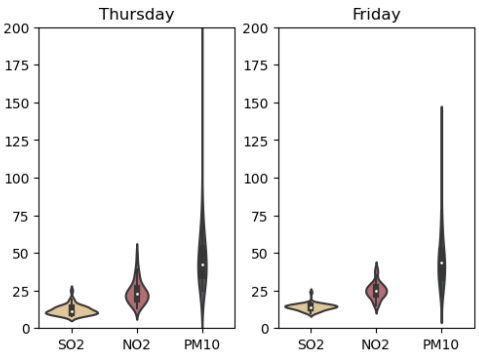
Friday NA



Residential, Rural and other Areas

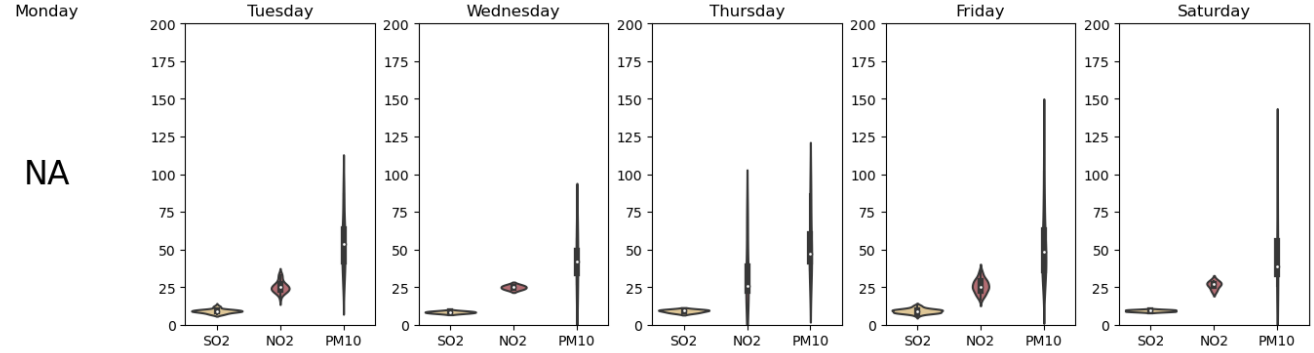


Wednesday NA

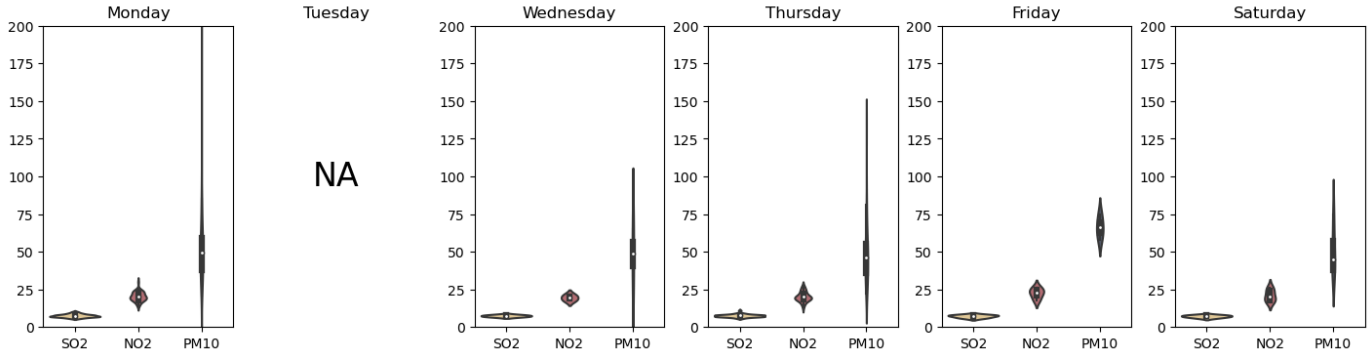


Saturday NA

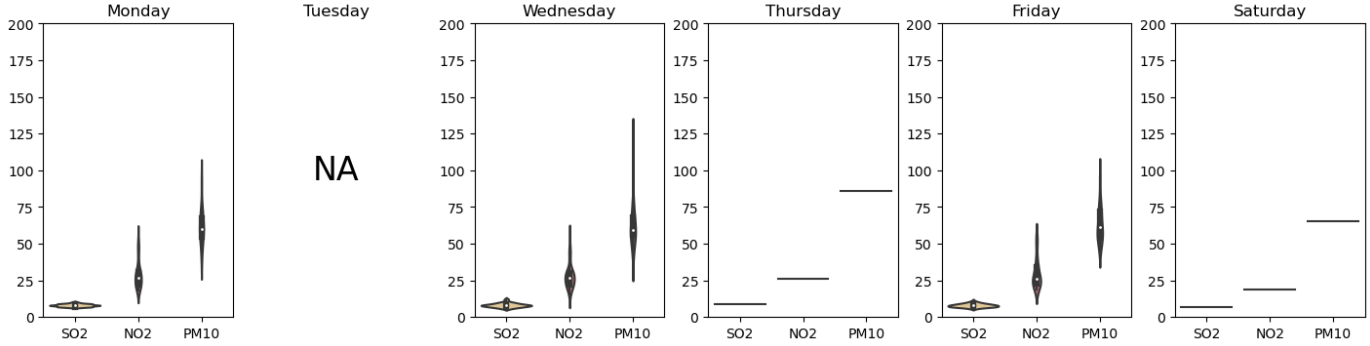
Mettur
Industrial Area



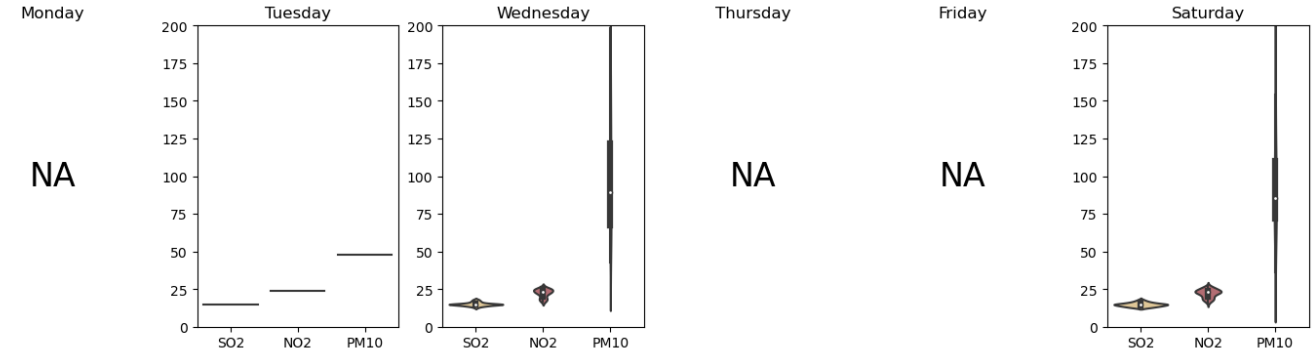
Residential, Rural and other Areas



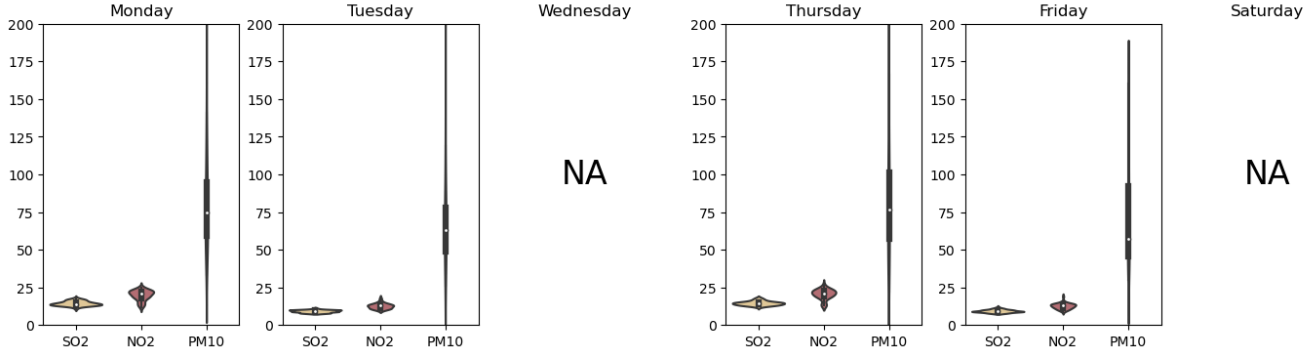
Salem
Residential, Rural and other Areas

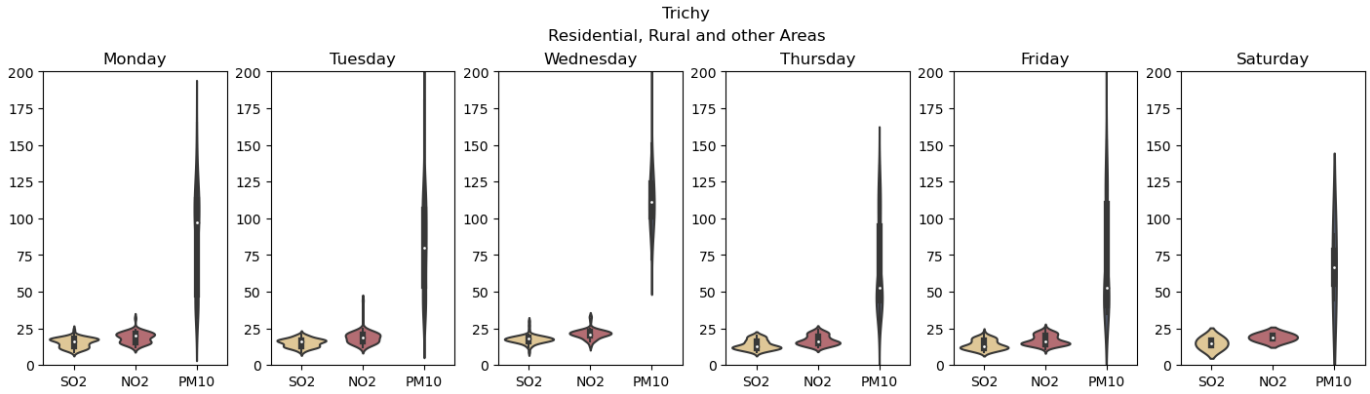


Thoothukudi
Industrial Area



Residential, Rural and other Areas



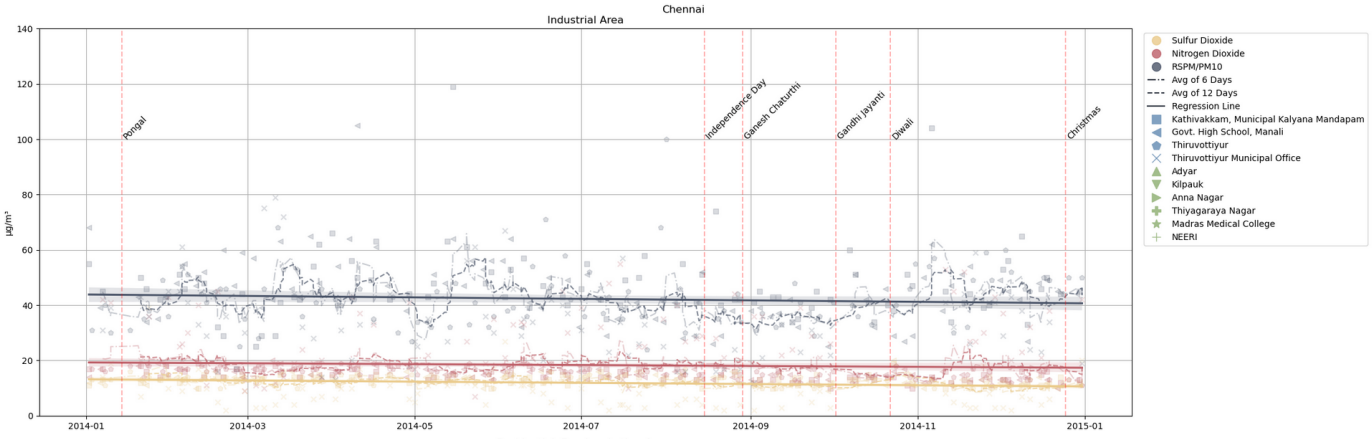


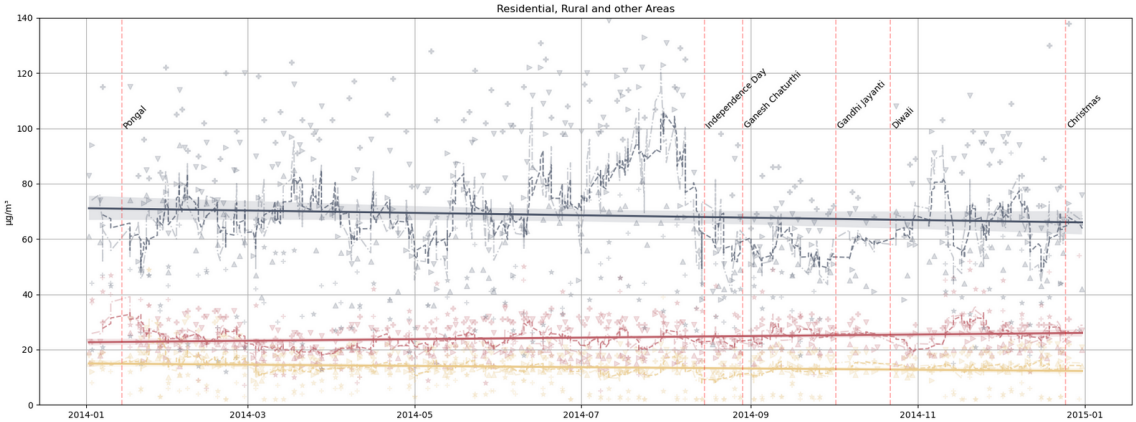
SO2, NO2, & PM10 in Every Area and Location Type, Regression Line, With Important Holidays Marked

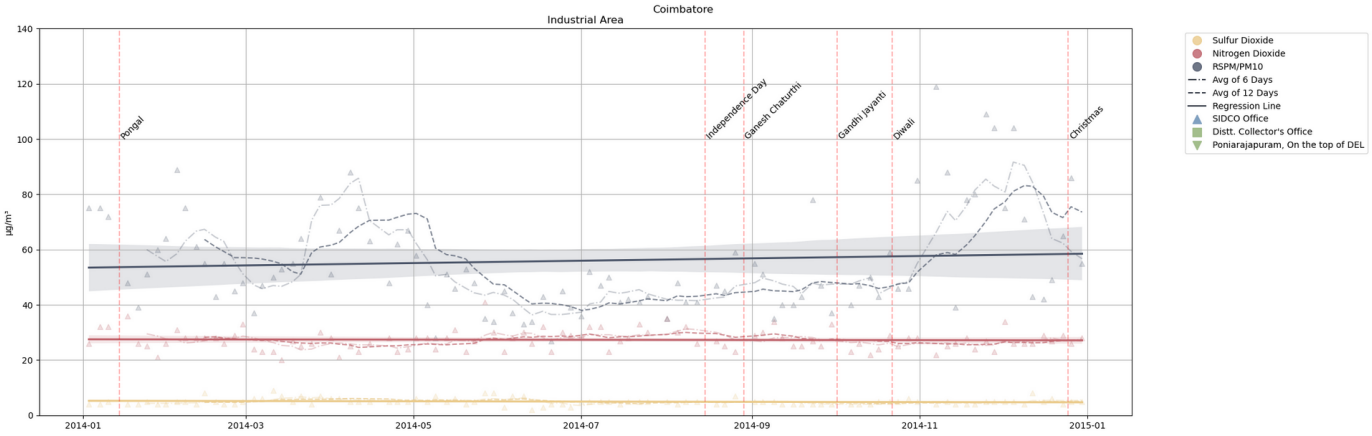
```
# ADD MORE IMPORTANT DATES
event_dates = {
    "Diwali": "2014-10-22",
    "Christmas": "2014-12-25",
    "Pongal": "2014-01-14",
    "Independence Day": "2014-08-15",
    "Ganesh Chaturthi": "2014-08-29",
    "Gandhi Jayanti": "2014-10-02",
}

def draw_vertical_lines(events,ax,y_cord=50):
    for key,val in events.items():
        x_val = datetime.strptime(val, "%Y-%m-%d")
        ax.axvline(x=x_val, color='red', linestyle='--',alpha=.3,)
        ax.text(x_val, y_cord, key, color='black',rotation=45)

markers = ['s', '^', 'v', '<', '>', 'p', 'P', '*', '+', 'x', 'D', 'd', 'H',
'h']

def draw_(area,mark_alpha = 0.2,d3_alpha=0.3,reg_alpha=1,d7_alpha=.7):
    fig, axes = plt.subplots(nrows=2,ncols =1,figsize=(22,14))
    stn_marker_map = {stn_name: markers[i] for i, stn_name in
enumerate(data_by_area[area]["loc"].unique())}
    station_name_legends = []
    for i, (name, data_) in
enumerate(data_by_area[area].groupby("loctype")):

        # Raw Data Points
        for stn_name in data_["loc"].unique():
            marker = stn_marker_map[stn_name]
            _data_ = data_[data_["loc"] == stn_name]
```

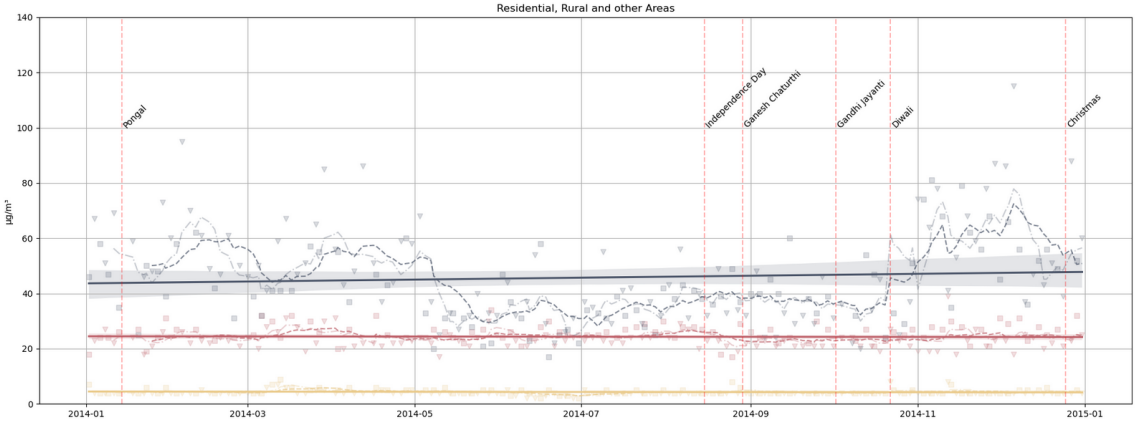
```
            stn_name = stn_name[:stn_name.rfind(",")]

axes[i].scatter(_data_.date,_data_.SO2,color=so2_color,alpha=mark_alpha,mark
er=marker)
            axes[i].scatter(_data_.date,_data_.NO2, color =
no2_color,alpha=mark_alpha,marker=marker)

axes[i].scatter(_data_.date,_data_["RSPM/PM10"],alpha=mark_alpha,marker=mark
er, color=pm10_color)
            station_name_legends.append(
                Line2D([0], [0], color="#81A1C1" if name == unique_loctyp[0]
else "#A3BE8C", marker=marker, linestyle='', markersize=10, label=stn_name)
            )

        # Average of 6 days
        axes[i].plot(data_['date'], data_['SO2'].rolling(window=6).mean(),
color=so2_color, alpha=d3_alpha,linestyle="-.")
        axes[i].plot(data_['date'], data_['NO2'].rolling(window=6).mean(),
color=no2_color, alpha=d3_alpha,linestyle="-.")

axes[i].plot(data_['date'],data_['RSPM/PM10'].rolling(window=6).mean(),
color=pm10_color, alpha=d3_alpha,linestyle="-.")

        # Average of 12 Days
        axes[i].plot(data_['date'], data_['SO2'].rolling(window=12).mean(),
color=so2_color, alpha=d7_alpha,linestyle="--")
        axes[i].plot(data_['date'], data_['NO2'].rolling(window=12).mean(),
color=no2_color, alpha=d7_alpha,linestyle="--")

axes[i].plot(data_['date'],data_['RSPM/PM10'].rolling(window=12).mean(),
color=pm10_color, alpha=d7_alpha,linestyle="--")

        # Regression Line
        sns.regplot(x=date2num(data_['date']),y=data_['SO2'],
ci=95,color=so2_color, line_kws={'alpha': reg_alpha},
scatter=False,ax=axes[i])
        sns.regplot(x=date2num(data_['date']),y=data_['NO2'],
ci=95,color=no2_color, line_kws={'alpha': reg_alpha},
scatter=False,ax=axes[i])
        sns.regplot(x=date2num(data_['date']),y=data_['RSPM/PM10'],
ci=95,color=pm10_color, line_kws={'alpha': reg_alpha},
scatter=False,ax=axes[i])
```

```
        axes[i].set_title(name)
        axes[i].set_ylabel("µg/m³")
        axes[i].set_ylim(0,140)
        axes[i].grid(True)
        draw_vertical_lines(event_dates,axes[i],100)

    if i != 1:
        axes[1].remove()

    common_color = "#2E3440"
    custom_legend = [
        Line2D([0], [0],alpha=0.8, color=so2_color, marker='o',
linestyle='', markersize=10, label='Sulfur Dioxide'),
        Line2D([0], [0],alpha=0.8, color=no2_color, marker='o',
linestyle='', markersize=10, label='Nitrogen Dioxide'),
        Line2D([0], [0],alpha=0.8, color=pm10_color, marker='o',
linestyle='', markersize=10, label='RSPM/PM10'),
        Line2D([0], [0], color=common_color, marker='', linestyle='-.',
markersize=10, label='Avg of 6 Days'),
        Line2D([0], [0], color=common_color, marker='', linestyle='--',
markersize=10, label='Avg of 12 Days'),
        Line2D([0], [0], color=common_color, marker='', linestyle='-',
markersize=10, label='Regression Line'),
        *station_name_legends
    ]

    axes[0].legend(loc='upper right', bbox_to_anchor=(1.22,
1),handles=custom_legend)
    plt.suptitle(area)
    plt.tight_layout()
    plt.show()

for area in data_by_area.keys():
    draw_(area)
```

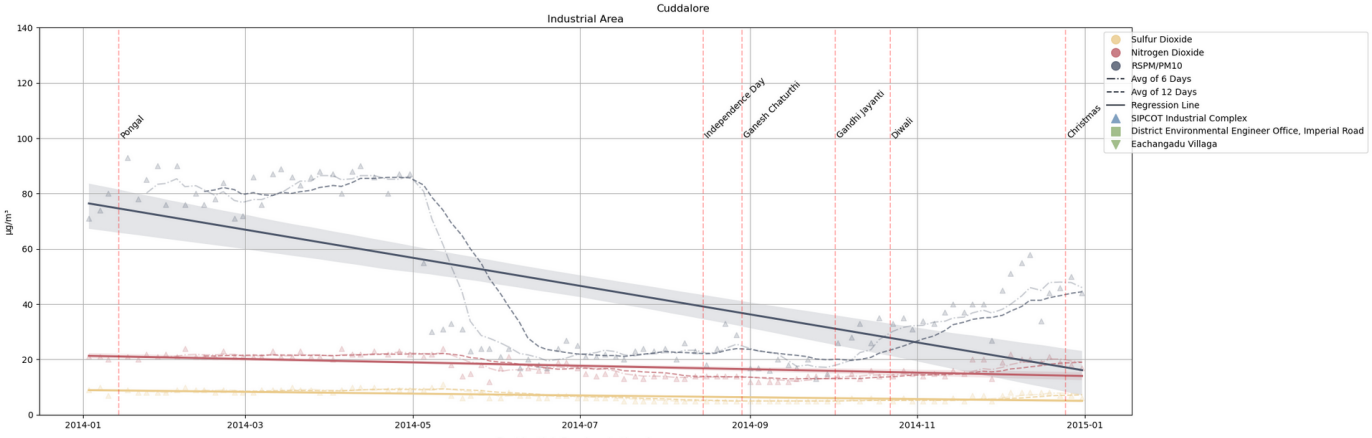

Chennai
Industrial Area
µg/m³
Pongal
Independence Day
Ganesh Chaturthi
Gandhi Jayanti
Diwali
Christmas
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Sulfur Dioxide
Nitrogen Dioxide
RSPM/PM10
Avg of 6 Days
Avg of 12 Days
Regression Line
Kathivakkam, Municipal Kalyana Mandapam
Govt. High School, Manali
Thiruvottiyur
Thiruvottiyur Municipal Office
Adyar
Kilpauk
Anna Nagar
Thiyagaraya Nagar
Madras Medical College
NEERI

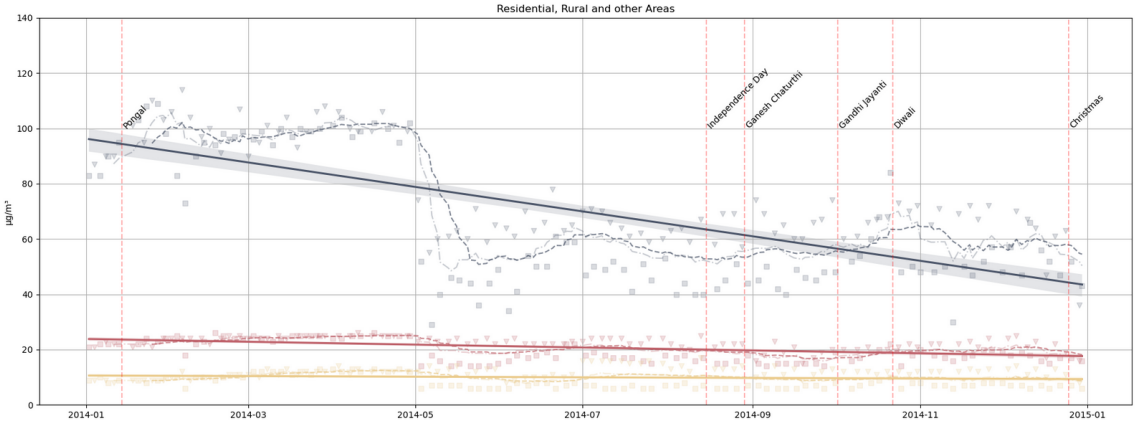

Residential, Rural and other Areas
µg/m³
Pongal
Independence Day
Ganesh Chaturthi
Gandhi Jayanti
Diwali
Christmas
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01


Coimbatore
Industrial Area
µg/m³
Pongal
Independence Day
Ganesh Chaturthi
Gandhi Jayanti
Diwali
Christmas
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Sulfur Dioxide
Nitrogen Dioxide
RSPM/PM10
Avg of 6 Days
Avg of 12 Days
Regression Line
SIDCO Office
Distt. Collector's Office
Poniarajapuram, On the top of DEL

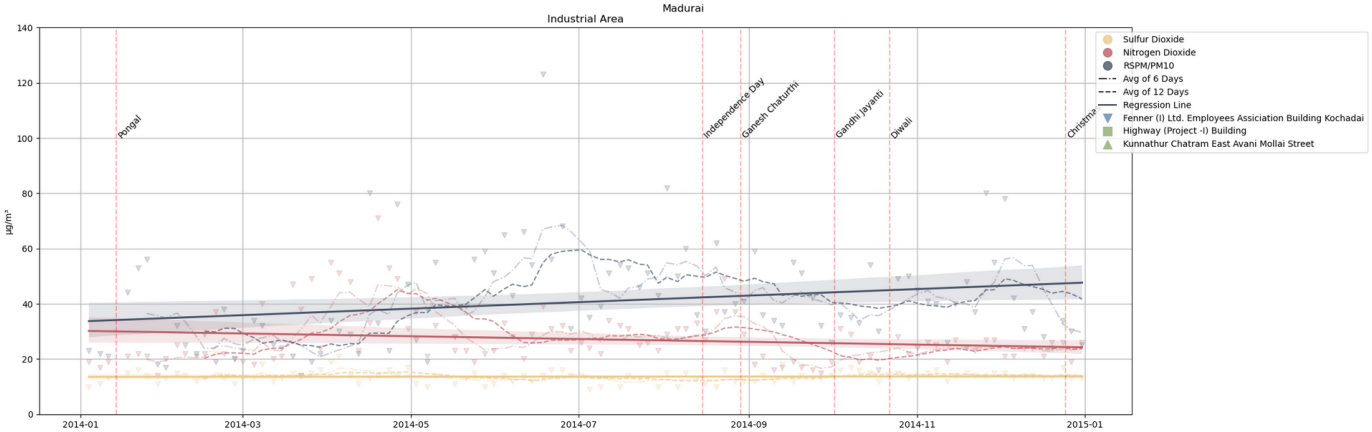

Residential, Rural and other Areas
µg/m³
Pongal
Independence Day
Ganesh Chaturthi
Gandhi Jayanti
Diwali
Christmas
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01

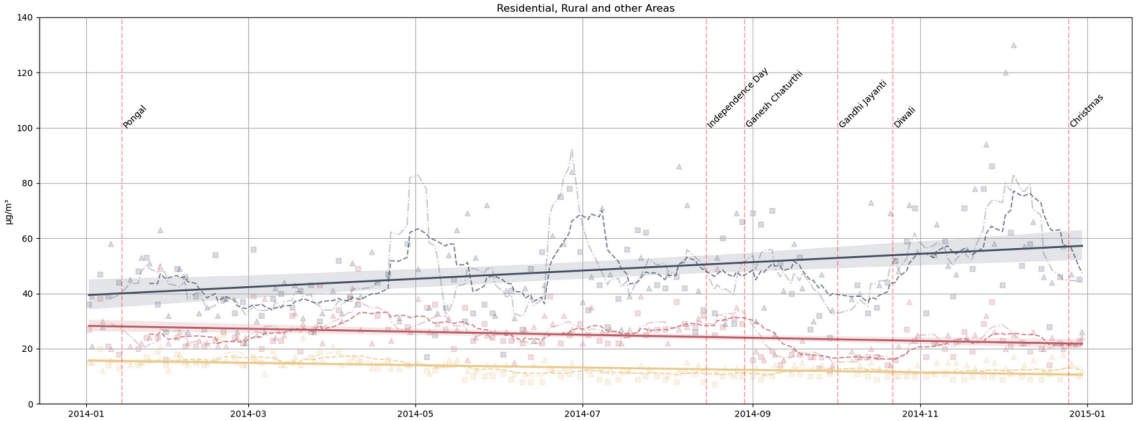
Cuddalore





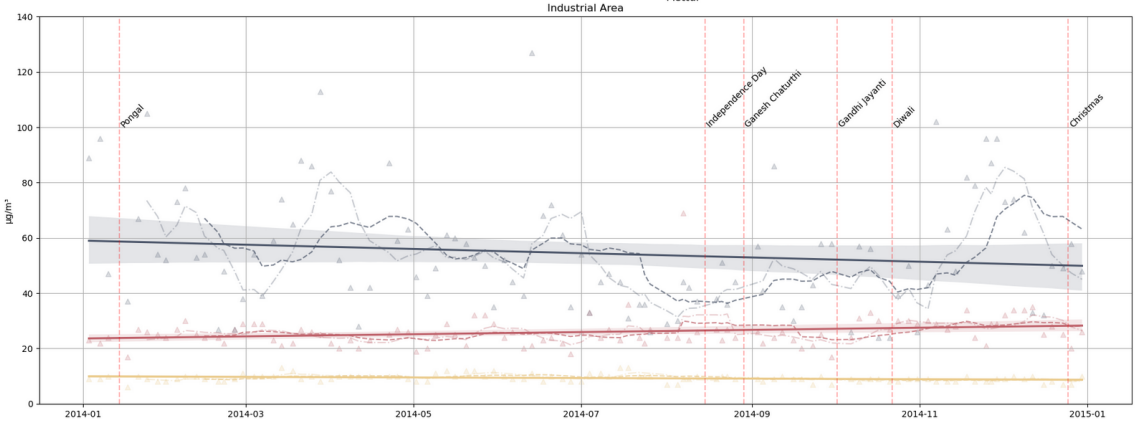
Madurai



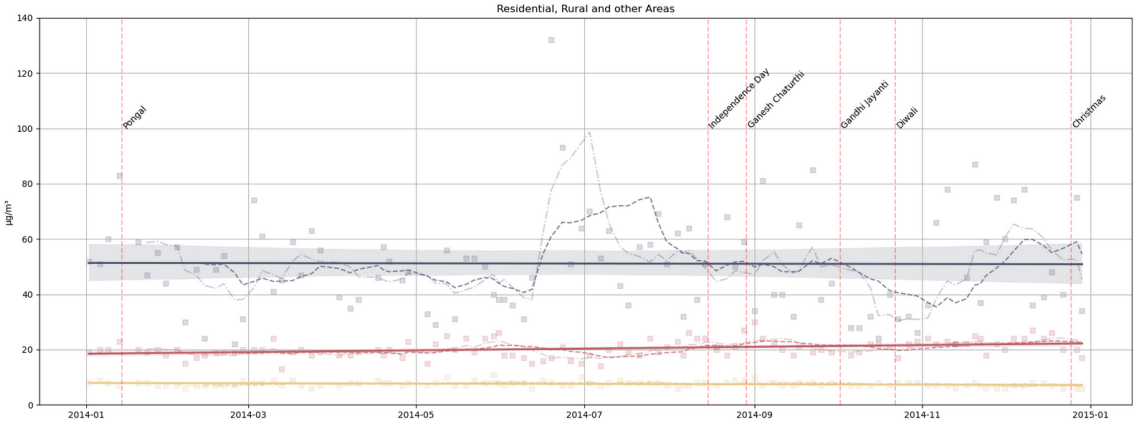


Mettur

Industrial Area

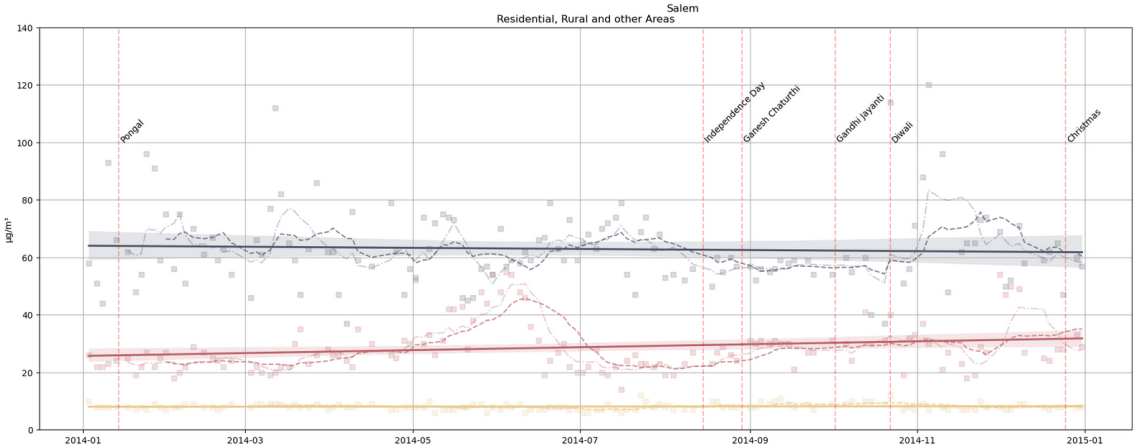


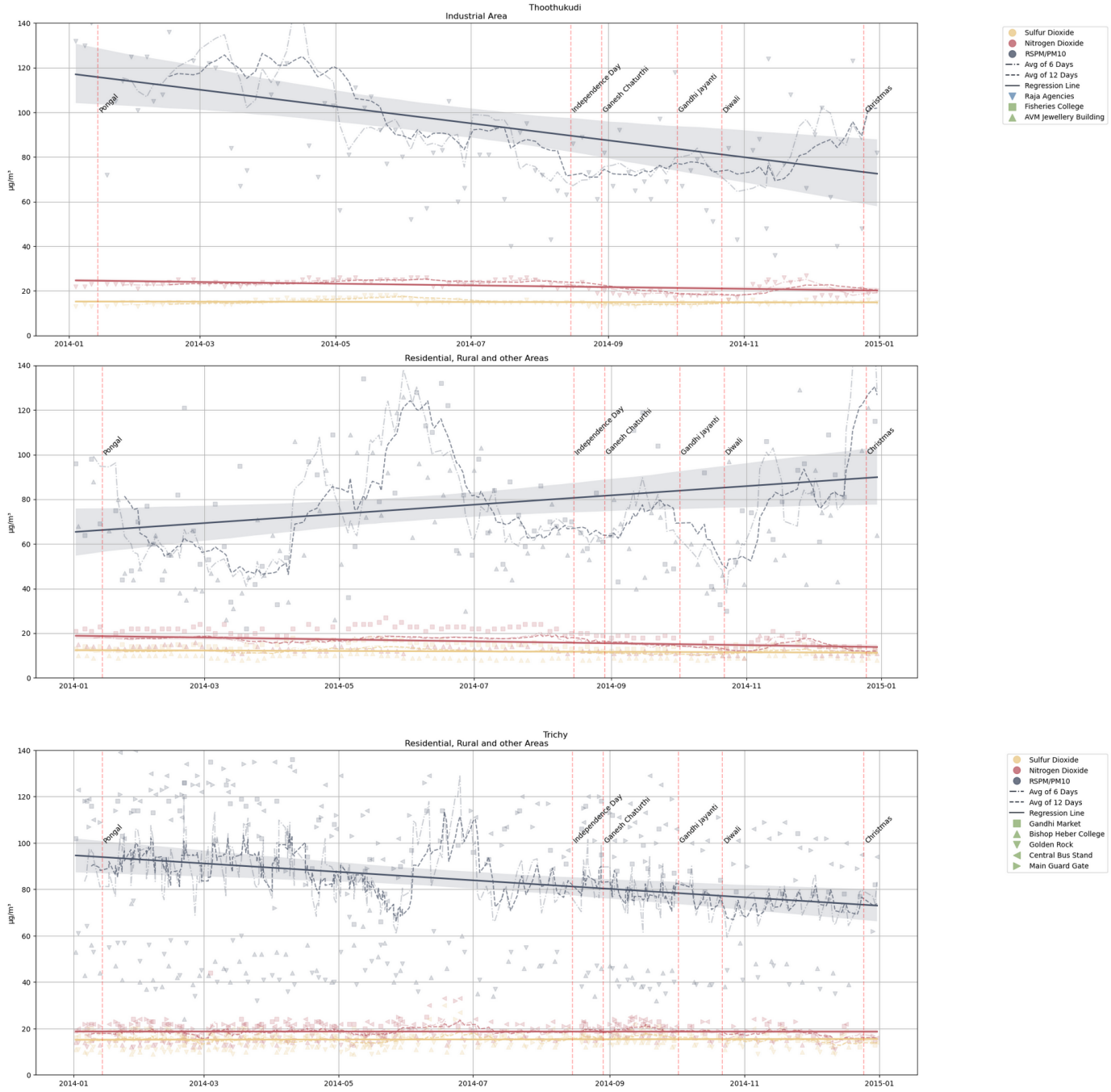
Residential, Rural and other Areas



Salem

Residential, Rural and other Areas





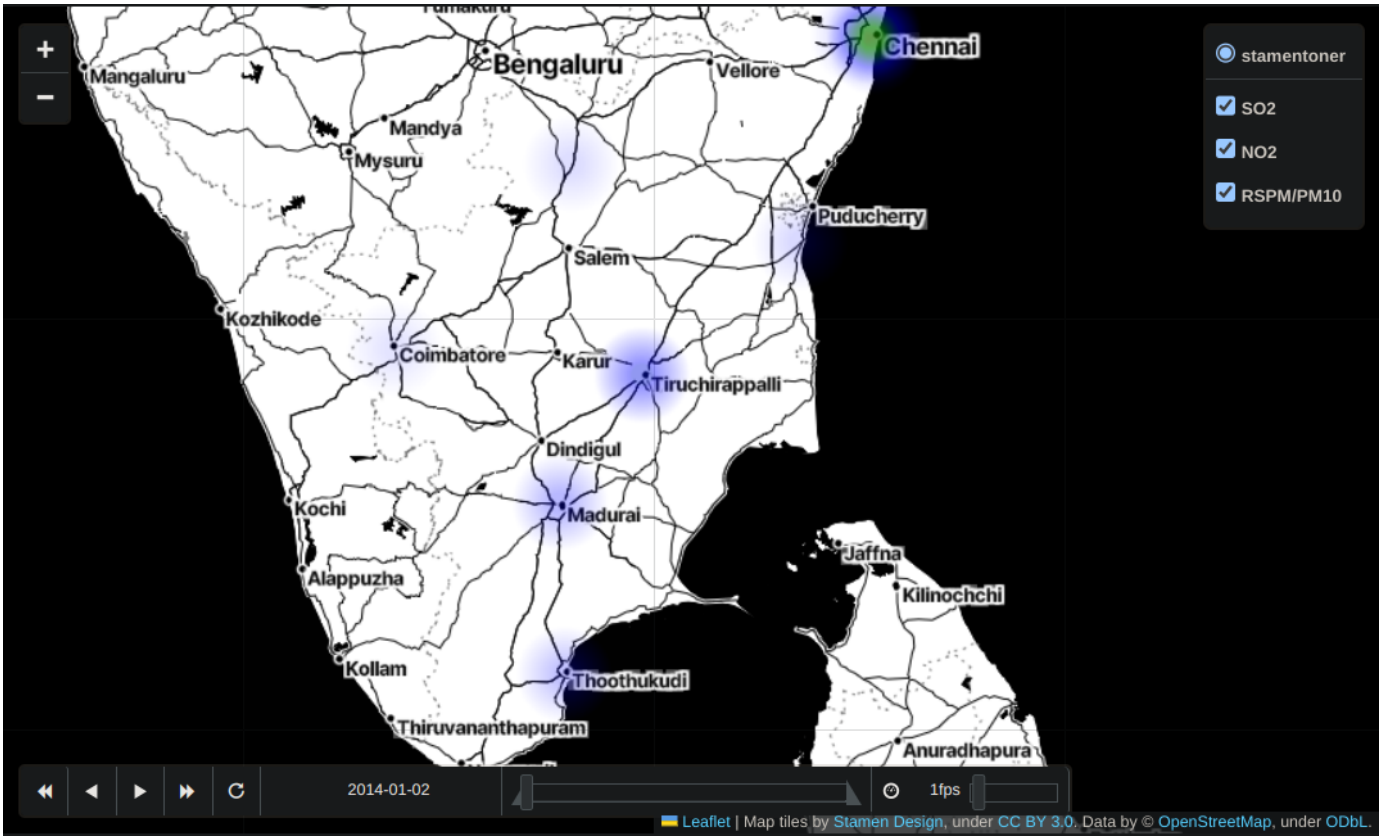
Geo Spatial Heatmap

```
def move_coordinates(lat, lon, min_dist=100, max_dist=5000):
    earth_radius = 6371000

    lat_rad = math.radians(lat)
    lon_rad = math.radians(lon)

    random_angle = random.uniform(0, 2 * math.pi)
    distance_meters = random.uniform(min_dist, max_dist)

    new_lat = lat_rad + (distance_meters / earth_radius) *
math.cos(random_angle)
    new_lon = lon_rad + (distance_meters / earth_radius) *
```

```
math.sin(random_angle)

    new_lat = math.degrees(new_lat)
    new_lon = math.degrees(new_lon)

    return new_lat, new_lon

def scale_01(curr, max_val, min_val):
    return (curr - min_val) / (max_val - min_val)

def create_data_for_heatmap(data,duplicates=0):
    station_cord_map = stn_df.set_index("name")
    max__ = desc.loc["max"]
    min__ = desc.loc["min"]
    so2_full = []
    no2_full = []
    pm10_full = []
    time_index = []
    for date, group in data.groupby("date"):
        time_index.append(date.strftime("%Y-%m-%d"))
        so2_data = []
        no2_data = []
        pm10_data = []
        for _,row in group.iterrows():
            lat, lon = station_cord_map.loc[row["loc"]]["coord"]
            SO2 = scale_01(row["SO2"],max__[0],min__[0])
            NO2 = scale_01(row["NO2"],max__[1],min__[1])
            PM10 = scale_01(row["RSPM/PM10"],max__[2],min__[2])
            for dup in range(duplicates+1):
                if dup == 0:
                    lat_,lon_ = lat, lon
                else:
                    lat_, lon_ = move_coordinates(lat,lon)
                so2_data.append(
                    [lat_,lon_,SO2]
                )
                no2_data.append(
                    [lat_,lon_,NO2]
                )
                pm10_data.append(
                    [lat_,lon_,PM10]
                )
        so2_full.append(so2_data)
```

```
        no2_full.append(no2_data)
        pm10_full.append(pm10_data)
    return so2_full,no2_full,pm10_full, time_index



A,B,C,time_index = create_data_for_heatmap(data,0)

class HeatMapWithTimeAdditional(Layer):
    _template = Template("""
        {% macro script(this, kwargs) %}
            var {{this.get_name()}} = new TDHeatmap({{ this.data }},
                {heatmapOptions: {
                    radius: {{this.radius}},
                    minOpacity: {{this.min_opacity}},
                    maxOpacity: {{this.max_opacity}},
                    scaleRadius: {{this.scale_radius}},
                    useLocalExtrema: {{this.use_local_extrema}},
                    defaultWeight: 1,
                    {% if this.gradient %}gradient: {{ this.gradient }}{%
endif %}
                }
            }).addTo({{ this._parent.get_name() }});
        {% endmacro %}
    """)

    def __init__(self, data, name=None, radius=15,
                 min_opacity=0, max_opacity=0.6,
                 scale_radius=False, gradient=None, use_local_extrema=False,
                 overlay=True, control=True, show=True):
        super(HeatMapWithTimeAdditional, self).__init__(
            name=name, overlay=overlay, control=control, show=show
        )
        self._name = 'HeatMap'
        self.data = data

        # Heatmap settings.
        self.radius = radius
        self.min_opacity = min_opacity
        self.max_opacity = max_opacity
        self.scale_radius = 'true' if scale_radius else 'false'
        self.use_local_extrema = 'true' if use_local_extrema else 'false'
        self.gradient = gradient
```

```
tn_map = folium.Map(location=[10.5, 79],tiles="stamentoner", zoom_start=7.4)

HeatMapWithTime(A,name="SO2",index=time_index,radius=30,max_opacity=0.9,grad
ient={0.4: 'blue', 0.65: 'green', 1: 'red'}).add_to(tn_map)
HeatMapWithTimeAdditional(B,name="NO2",radius=30,max_opacity=0.9,gradient=
{0.4: 'purple', 0.65: 'orange', 1: 'yellow'}).add_to(tn_map)
HeatMapWithTimeAdditional(C,name="RSPM/PM10",radius=30,max_opacity=0.9,gradi
ent={0.4: 'cyan', 0.65: 'magenta', 1: 'black'}).add_to(tn_map)


folium.LayerControl(collapsed=False).add_to(tn_map)
tn_map.save('geo_spatial_aq_heatmap.html')
tn_map
```

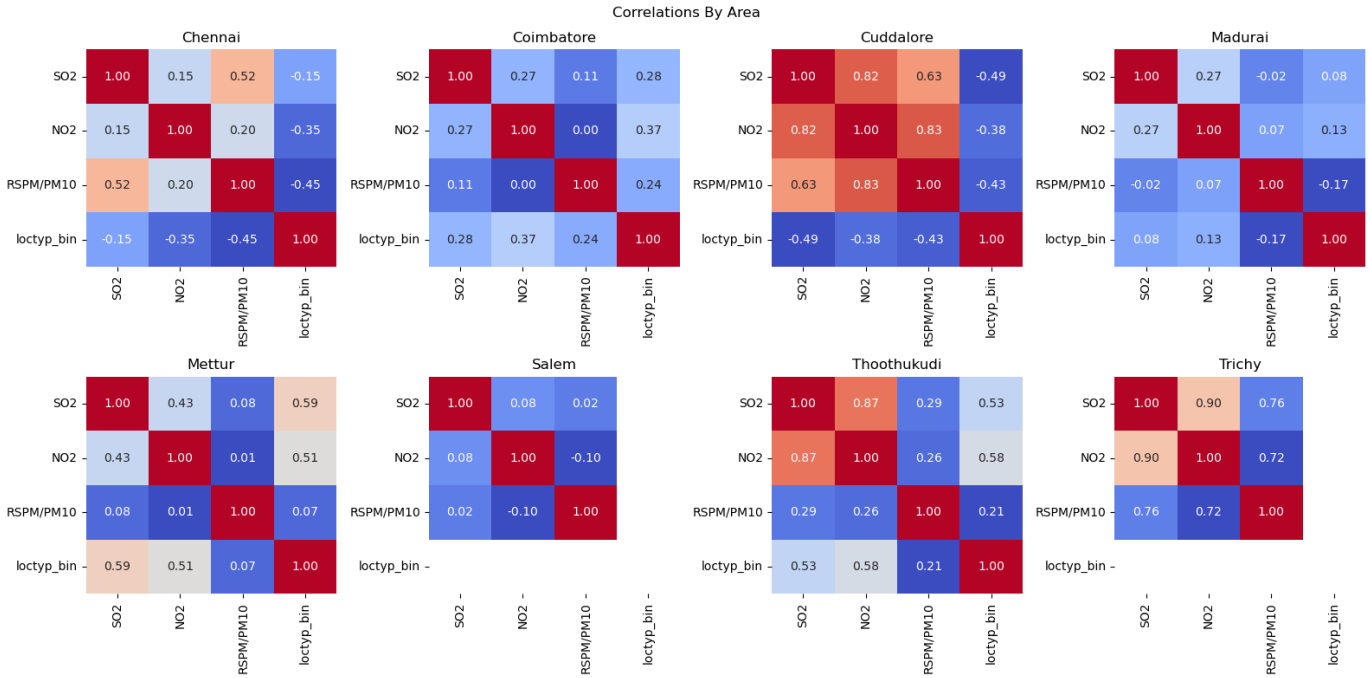


link to the html file, geo_spatial_aq_heatmap.html

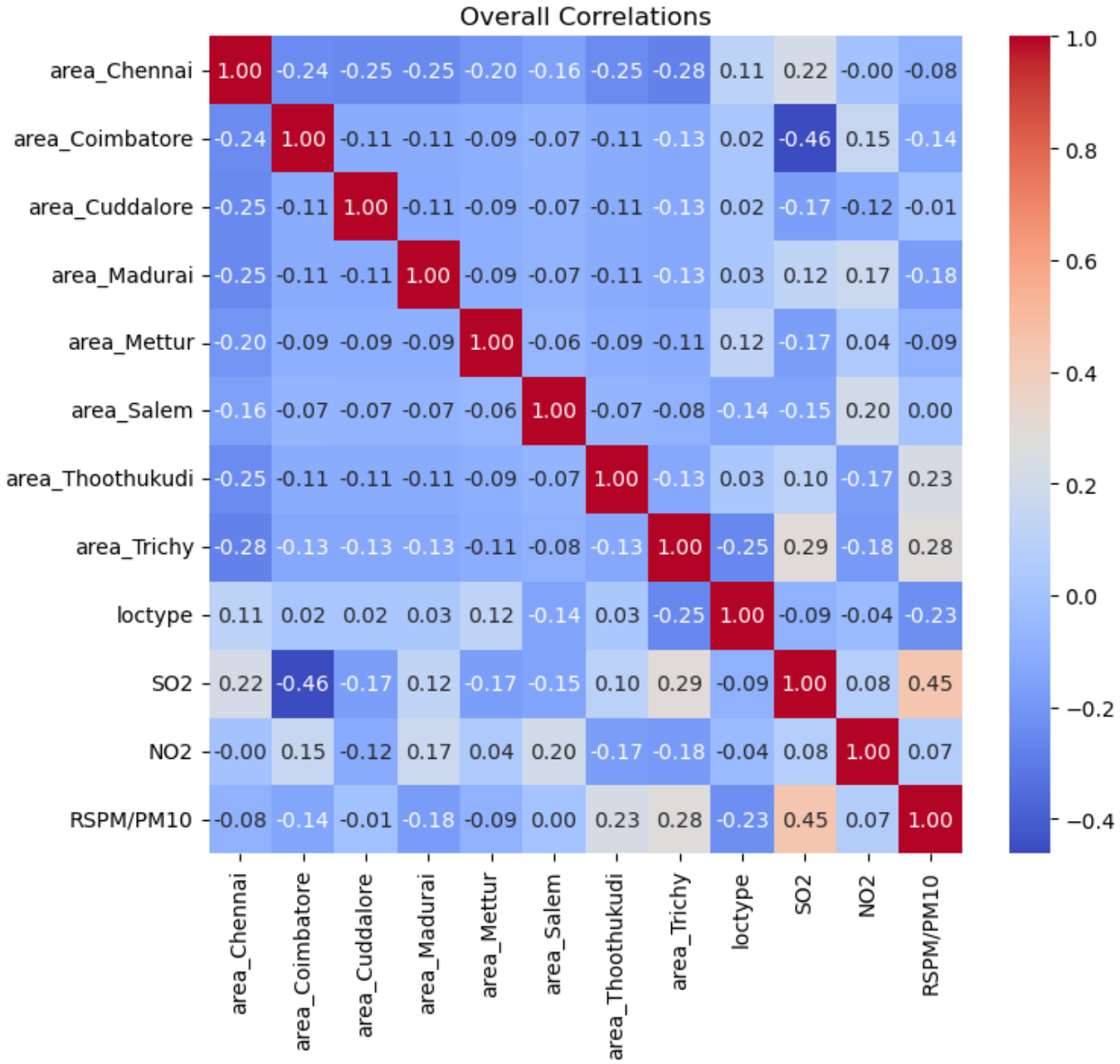
Heat Map of Correlations between SO2, NO2, Location Type & PM10.

```
fig, axes = plt.subplots(2, 4, figsize=(16, 8))
fig.subplots_adjust(hspace=0.5)

for i, area in enumerate(unique_areas):
    row, col = divmod(i, 4)  # Calculate the row and column for the current
area
    ax = axes[row, col]  # Select the current subplot
    temp = data_by_area[area]
```

```
    if "loctyp_bin" not in data.columns:
        temp["loctyp_bin"] = temp["loctype"].apply(lambda x: 1 if x[0] ==
"I" else 0)
    corr_matrix = temp.corr(numeric_only=True)

    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0, cbar=False, ax=ax)
    del temp
    ax.set_title(area)
plt.suptitle("Correlations By Area")
plt.tight_layout()
plt.show()
```




```
data["loctyp_bin"] = data["loctype"].apply(lambda x: 1 if x[0] == "I" else
0)

X = pd.get_dummies(data['area'], prefix='area')
X["loctype"] = data["loctyp_bin"]
X["SO2"] = data["SO2"]
X["NO2"] = data["NO2"]
X["RSPM/PM10"] = data["RSPM/PM10"]
fig = plt.figure(figsize=(8,7))
corr_matrix = X.corr(numeric_only=True)
del X
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=0)
plt.title("Overall Correlations")
plt.show()
```


Overall Correlations
area_Chennai 1.00 -0.24 -0.25 -0.25 -0.20 -0.16 -0.25 -0.28 0.11 0.22 -0.00 -0.08
area_Coimbatore -0.24 1.00 -0.11 -0.11 -0.09 -0.07 -0.11 -0.13 0.02 -0.46 0.15 -0.14
area_Cuddalore -0.25 -0.11 1.00 -0.11 -0.09 -0.07 -0.11 -0.13 0.02 -0.17 -0.12 -0.01
area_Madurai -0.25 -0.11 -0.11 1.00 -0.09 -0.07 -0.11 -0.13 0.03 0.12 0.17 -0.18
area_Mettur -0.20 -0.09 -0.09 -0.09 1.00 -0.06 -0.09 -0.11 0.12 -0.17 0.04 -0.09
area_Salem -0.16 -0.07 -0.07 -0.07 -0.06 1.00 -0.07 -0.08 -0.14 -0.15 0.20 0.00
area_Thoothukudi -0.25 -0.11 -0.11 -0.11 -0.09 -0.07 1.00 -0.13 0.03 0.10 -0.17 0.23
area_Trichy -0.28 -0.13 -0.13 -0.13 -0.11 -0.08 -0.13 1.00 -0.25 0.29 -0.18 0.28
loctype 0.11 0.02 0.02 0.03 0.12 -0.14 0.03 -0.25 1.00 -0.09 -0.04 -0.23
SO2 0.22 -0.46 -0.17 0.12 -0.17 -0.15 0.10 0.29 -0.09 1.00 0.08 0.45
NO2 -0.00 0.15 -0.12 0.17 0.04 0.20 -0.17 -0.18 -0.04 0.08 1.00 0.07
RSPM/PM10 -0.08 -0.14 -0.01 -0.18 -0.09 0.00 0.23 0.28 -0.23 0.45 0.07 1.00
area_Chennai
area_Coimbatore
area_Cuddalore
area_Madurai
area_Mettur
area_Salem
area_Thoothukudi
area_Trichy
loctype
SO2
NO2
RSPM/PM10
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4