Impact Factor: 9.9 ISSN-L: 2544-980X

Analysis of Air Pollution Levels Using Python Libraries

Jurayev Jakhongir [1], *Mekhmonaliyev Yakhyobek* [2]

Annotation: This article aims to help those who are facing the same issues as me: working with Python but missing the OpenAir package of R.

Keywords: Raster data structure, vector data structure, basic spatial data analysis, encoding data structure, topological data structure.

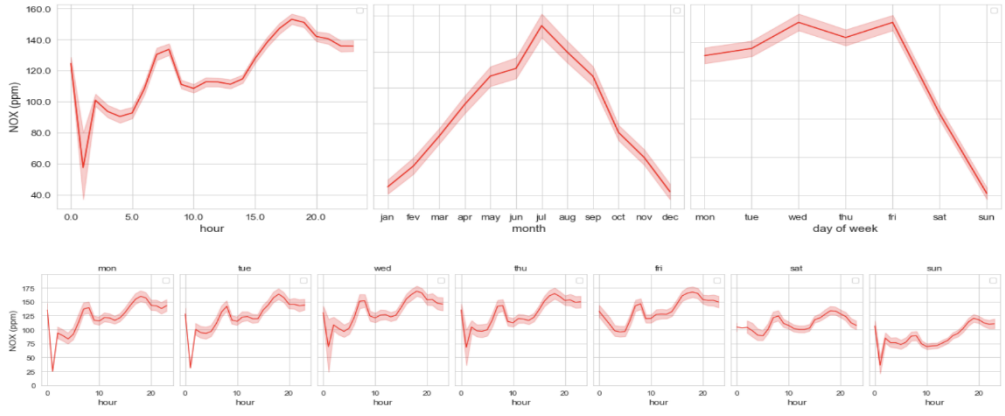
Recently, much has been discussed about air pollution and its consequences on the environment. These discussions always gain prominence when some of their consequences haunt the world and leave us wondering what will be of future generations. Maybe greenhouse gases and the consequent global warming are the most well-known names for most of the population. However, other air pollutants cause many harmful effects to the population and end up not being well regulated in many regions. These are the cases of particulate matter and nitrogen oxides, for example. In large cities, these pollutants are usually emitted on a large scale by diesel vehicles that lack proper maintenance. For a more assertive assessment of the state of pollutant concentrations in a given region, an in-depth study of the data is necessary. For that, a couple of years ago we started using the **OpenAir** package of R, which by the way is excellent. However, over the last few months, We've started using Python and we've come to like it. But what we miss the most about R is OpenAir.

With that in mind, we decided to develop **two functions in Python**, for visualizing pollutants datasets, **inspired by OpenAir of R**. So, in this article, we'll present these functions and we'll detail how they were developed.

It's important to make it clear that we're not going to deal with the analysis of local pollutants here, but rather use the data as an example to illustrate our analysis tools. By the way, we used a dataset from: https://qualar.cetesb.sp.gov.br/qualar/home.do, which is the agency responsible for regulating environmental issues in the state of Sao Paulo.

What are the functions we've developed?

The first one is a set of plots, which we called **time_variation**, showing the concentration of the pollutant by the hour, month, and day of the week, like the image below.



[1] Ferghana branch of the Tashkent University of Information Technologies named after Muhammad al-Khorezmi

[2] Ferghana branch of the Tashkent University of Information Technologies named after Muhammad al-Khorezmi

Fig-1 The second one is a calendar heatmap (**calendar_dist**), which points the worst days relating to pollutant concentrations:

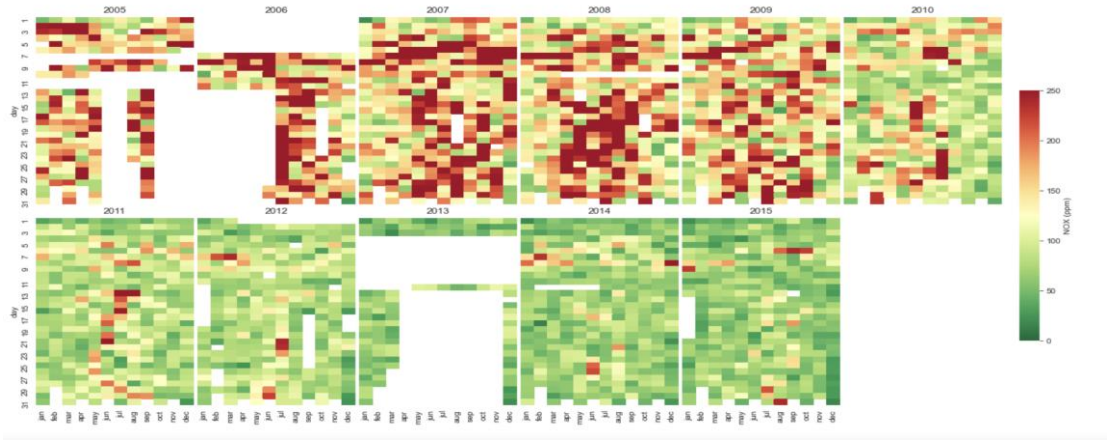


Fig-2

First things first. To kick off our function, we set the input variables (df, pollutant, ylabel, hue=None), and also, we have to import the libraries.

```
def time_variation(df, pollutant, ylabel, hue=None):
# importing all the libraries we'll need
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

As we're going to use int values representing months and days of the week to create the charts, we need lists of labels to replace these int values in the plots.

```
#setting xticklabels
week = ['mon','tue','wed','thu','fri','sat','sun']
months = ['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']
```

To make our function work fine, we need to be sure the data in feature date is really in date format. So:

```
# converting feature date to datetime and dropping Nan values
df['date'] = pd.to_datetime(df['date'])
if hue!=None:
df = df[['date', pollutant, hue]].dropna() # dropping rows of nan vals of the chosen pollutant
else:
df = df[['date', pollutant]].dropna()
```

Now, we only need to plot. But wait, first let's set the subplots. There will always be **3 subplots**, representing concentrations by the hour, month, and day of the week.

```
# Creating graphs of pollutant concentrations by hour, month and day of week
fig,axes = plt.subplots(1, 3,sharex=False, figsize=(16,4)) #creating subplots, side by side
fig.tight_layout(pad=-2) # makeing plots get closer
```

sns.set_style('whitegrid')

After that, plotting the graphs is very simple. In the example below, we show the lineplot of *concentration by the hour*. The others are very similar, we just have to change hours by month or day of week.

```
# concentration vs hour
axes[0] = sns.lineplot(ax=axes[0],data=df,
x=df['date'].dt.hour,
y=pollutant,
color='red',
linewidth=1.5,
hue=hue,
palette="hls")
axes[0].set_xticklabels(axes[0].get_xticks(), fontsize=13)
axes[0].set_yticklabels(axes[0].get_yticks(), fontsize=13)
axes[0].set_xlabel('hour', fontsize=15)
axes[0].set_ylabel(ylabel, fontsize=15)
axes[0].legend().set_title('')

# concentration vs month
axes[1] = sns.lineplot(ax=axes[1],
data=df,
x=df['date'].dt.month,
y=pollutant,
color='red',
linewidth=1.5,
hue=hue,
palette="hls")
axes[1].set_xticks(np.arange(1, 13, 1))
axes[1].set_xticklabels(months, fontsize=13)
axes[1].set_yticklabels('')
axes[1].set_xlabel('month', fontsize=15)
axes[1].set_ylabel('')
axes[1].legend().set_title('')

# concentration vs day of week
axes[2] = sns.lineplot(ax=axes[2],
data=df,
x=df['date'].dt.dayofweek,
```

```
y=pollutant,
color='red',
linewidth=1.5,
hue=hue,
palette="hls")
axes[2].set_xticks(np.arange(0, 7, 1))
axes[2].set_xticklabels(week, fontsize=13)
axes[2].set_yticklabels('')
axes[2].set_xlabel('day of week', fontsize=15)
axes[2].set_ylabel('')
axes[2].legend().set_title('')
```

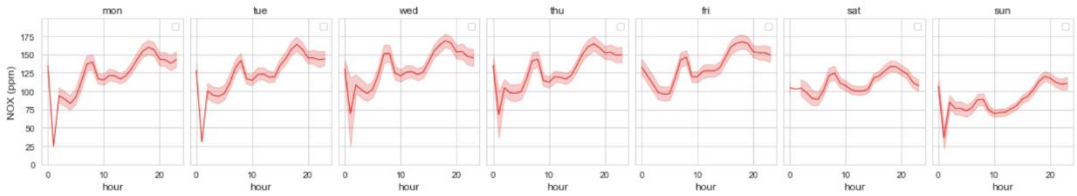


Fig-3

In the second part of our Time Variation, we want to check if there is any **difference between mean pollutant concentrations by the hour and by days of the week**. This is a great tool to use for checking the seasonality of a pollutant during a week. In the example above, it's clear that the pollutant concentrations decrease on weekends, which can be related to the reduction of car traffic on these days.

We hope this article can help many others who are working on this topic of research, and also we hope you enjoy it as we did.

List of reference:

1. Qadamova, Z., & Sotvoldiyeva, N. (2023). DASTURLASHNI O 'RGANUVCHILAR UCHUN ENG YAXSHI DASTURLASH TILLARI. Educational Research in Universal Sciences, 2(10), 241-244.
2. Хусанова, М. К., & Сотволдиева, Д. Б. (2020). ИСПОЛЬЗОВАНИЕ ДЕЦИМАЦИИ И ИНТЕРПОЛЯЦИИ ПРИ ОБРАБОТКЕ СИГНАЛОВ В ПРОГРАММЕ MATLAB. In ЦИФРОВОЙ РЕГИОН: ОПЫТ, КОМПЕТЕНЦИИ, ПРОЕКТЫ (pp. 970-975).
3. Toxirova, S. (2023). Python dasturida lug'atlar bilan ishlash . Conference on Digital Innovation : "Modern Problems and Solutions". извлечено от https://fer-teach.uz/index.php/codimpas/article/view/1910
4. Muhammadjonov, A., & TURLARI, T. S. Y. T. ICHKI VA TASHQI YARIMO 'TKAZGICHLAR. Research and implementation.–2023.20, 23.
5. D. Sotvoldieva. (2023). FREQUENCY ANALYSIS OF THE SIGNAL. Best Journal of Innovation in Science, Research and Development, 2(11), 693–699. Retrieved from