Introduction

Data science is a powerful tool used by researchers in various fields to test scientific hypothesises. It is a multi-step process which involves pre-processing large data samples, analysing the data samples with various regression tools and then visualizing the results. Here we will go through an example case of using data science to see the effect of COVID-19 pandemic on the quality of air we breathe.



Pictures taken over Milan, Italy, before (left) and after (after) COVID-19 pandemic

The COVID-19 pandemic started affecting the world in early 2020. Various countries started implementing lockdown measures to decelerate the spread of this pandemic. The lockdown measures involved restrictions on unessential travels. Travel vehicles emit various pollutant gases such as nitrogen dioxide (NO2) which directly affect the quality of air we breathe (forming harmful particles) and our health (inflammation of airways). Here we will use data science capability of Python to see the effect of the travel restrictions on NO2 levels in Bristol and London, two largest cities in southern England.

The Department of Environment Food and Rural Affairs (DEFRA) website (<https://uk-air.defra.gov.uk/>) provides free access to measurement data of various pollutants at various sites within the UK. A csv file containing NO2 measurements from the sites at Bristol Temple Way and London Marylebone Road is provided in the resources folder for this project. Detail instructions on accessing data from the DEFRA website are provided in the Primary Science Teaching Trust website (<https://pstt.org.uk/resources/curriculum-materials/post-16-citizen-science-air-pollution>).

Step 1: Getting Started with Jupyter

Jupyter is an interactive development environment for Python and we will use it to write and execute our Python program for data analysis. Before using Jupyter, create a folder named AirWeBreathe inside your named folder to save all the files for this project. Jupyter will open in an internet browser and you should browse to the AirWeBreathe folder we created earlier. After you are inside your AirWeBreathe folder which should be empty, upload the AirWeBreathe.csv file from the python resource folder using the upload button on the right-hand corner. After upload you should see the csv file.



Create a new Python 3 notebook using the dropdown option under New. This will open a new tab in the browser and a Jupyter notebook titled Untitled will be created. Go back to the Jupyter tab and select the Untitled notebook and rename (under Files option) it to AirWeBreathe.ipynb



Step 2: Importing Modules and Data

Now we will import numpy and pandas which are very popular modules for data analysis in Python. NumPy is a standard module in Python whereas you may need to install the pandas module manually before importing. Shown below is a link to pandas manual and installation guide:

<https://pandas.pydata.org/pandas-docs/version/0.23.3/install.html>

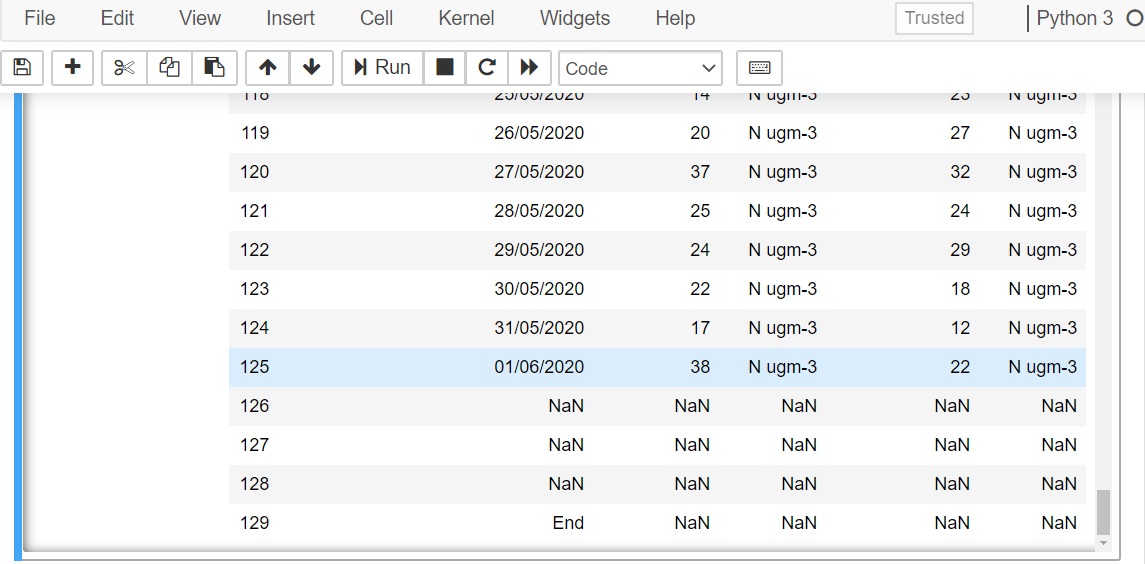
The empty cell (highlighted green when selected) takes the python commands and once you have typed the import commands execute it by using shift + return.



The pandas module has read\_csv function which we will use to import the NO2 measurement data from the csv files into Python as a dataframe, df. We will also display the contents of the measurement file using display command. As the file has quite a lot of rows we need to change the setting of the display command in the second line of code to 1000.



Now we can see the structure of our dataset collected from DEFRA. We can see the first column contains index number for all the dataset starting from 0. The first four rows only contain description of the data. After that we can see the columns are divided into Date, NO2 measurement from Bristol Temple Way, Measurement Units, NO2measurement from London Marylebone Road and Measurement Units. There is also a scroll bar on the right which we can use to explore all the rows of the dataset. Shown below are the last rows of the dataset.



We can see that the first four rows of the file do not actually contain measurement data. As part of data pre-processing, a new dataframe, sf, is created in the first three lines of code below. The dropna() function drops any rows which have one or more empty values. This is critical for large datasets which may have random empty spaces.The i and f values inside the square bracket provide the range of index to be copied into the new dataframe. The fourth line re-indexes the dataframe so that it starts from 0 again.



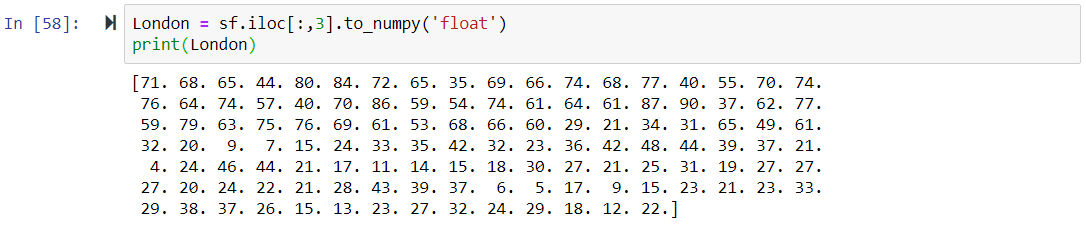
**Challenge Time!**

**Verify that the sf dataframe is indexed properly**

**using the display command**

Step 3: Data Slicing

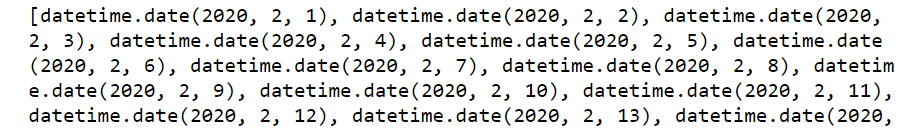
Now we will slice the imported dataframe into separate arrays for dates and measurements from London. Array in Python is a collection of data of the same type. First the measurement data from London is sliced and saved to an array named ‘London’. The iloc attribute of dataframe is used to select columns with index 1 and 3 for Bristol and London, respectively. The resulting dataframe series is then converted into NumPy ‘float type’ array using to\_numpy command. This conversion will enable us to use various NumPy functions for further analysis of data later. We can see the values stored in the ‘London’ array using the print command



A similar procedure is applied to extract dates for the measurement. Unlike the measurement values, dates are provided in a different format. We will import the datetime module to change the dates into a format readable in python. The third line in the code below uses list comprehension in which every value stored in the dates series is operated by the datetime commands.



Shown below is the output of the print command



**Challenge Time!**

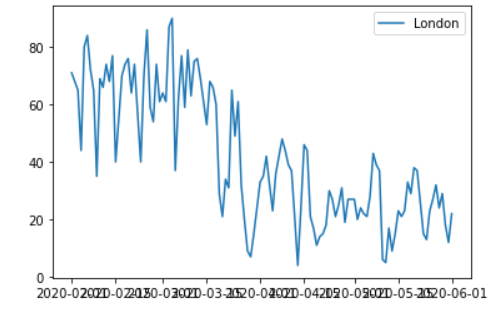
**Slice the Bristol measurements from sf dataframe**

**and save it to an array named ‘Bristol’**

Step 4: Data Visualisation

We will now plot these arrays to visualise the measurements from London. We will import matplotlib module for this as shown in the first line below. A figure object, fig, is created and subplot, ax, is added to it. A plot labelled ‘Bristol’ is created in the ax subplot to visualise the Bristol NO2 measurements. The plot and legend are displayed using the legend and show attributes of the matplotlib module

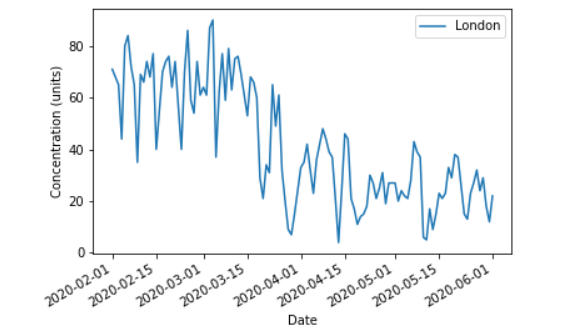




Further plot commands are added to display labels for the x (horizontal, xlabel) and y (vertical, ylabel) axis. The dates in the x axis is formatted for better visualisation using commands shown in line 7.



The output plot is show below which is much easier to read now!



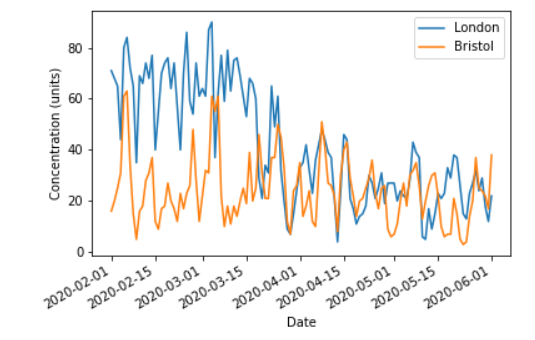
**Science Alert!**

**The concentration units are microgram cubic meter air**

**(μg m-3), 1 μg m-3 ~ 1 molecule per billion air molecules molecules**

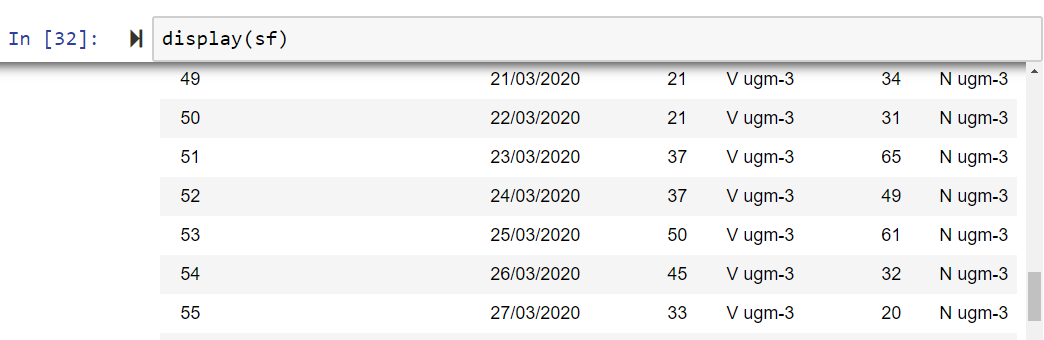
**Challenge Time!**

**Add a plot in the ax subplot to visualise the measurements from both London and Bristol. The final plot should look as shown below:**

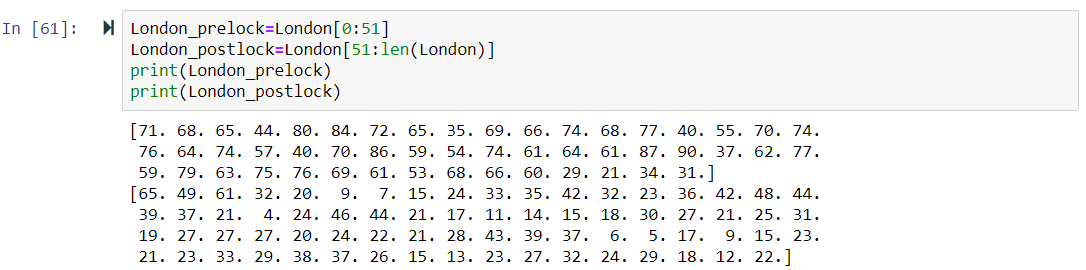


Step 5: Data Analysis

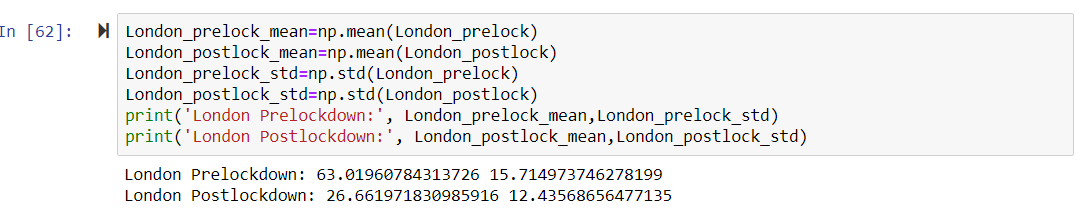
We can see that the NO2 concentration has gone down in London starting in April, whereas Bristol measurements have been relatively constant over the four-month time frame. To quantify the change in NO2 concentration we can average the concentration before and after the lockdown measurements were enforced by the UK government. The lockdown started on the evening of 23rd March and displaying the sf dataframe shows that this date corresponds to index 51.



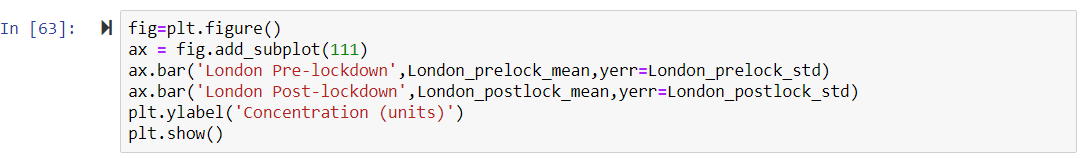
The ‘London’ array is split from index 51 into two separate arrays named ‘London\_prelock’ and ‘London\_postlock’. The print command is then used to confirm the split.



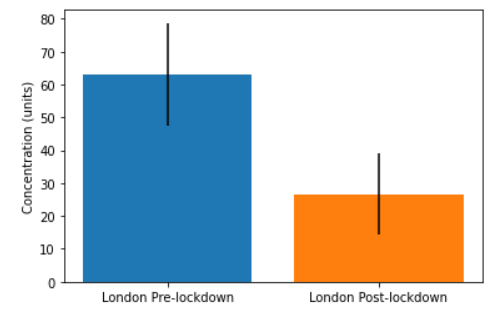
We can use mean and std function of the numpy modules to take average and standard deviation values of the measurement values in the two split arrays. The standard deviation represents average fluctuation in the daily measurements.



Finally, we can visualise these results in a bar plot to better summarise the results.



Shown below is the bar graph



The black vertical lines in the bar plot represent the standard deviations in NO2 measurements. The plot clearly shows that the mean NO2 concentration has decreased after March 23. Moreover, even the upper limit value for the post lockdown measurements, considering fluctuations in daily measurements, never reach the lower limit value for the pre-lockdown measurements [i.e. the two black vertical lines do not overlap vertically]. Thus, we can safely conclude that the NO2 measurement at the London site has decreased significantly due to the lockdown measures!

**Science Alert!**

**The NO2 concentration in London has decreased to within the recommended upper limit value of 40 μg m-3 set by the World Health Organisation!**

**Challenge Time!**

**Find whether NO2 concentration in Bristol has changed significantly due to the lockdown measures**

**Fun Challenge I!**

**Find which day of the week had the highest and lowest NO2 concentrations on average in Bristol**

**Fun Challenge II!**

**Using the AirWeBreathe\_seasonal.csv dataset (2019-2020 measurements) find whether the NO2 concentration changes in Bristol and London are seasonal or due to the lockdown measures**

This project guide was created by Dr. Rabi Chhantyal Pun from the School of Chemistry at University of Bristol as a part of the pathways to impact for the NERC grant NE/P013104/1