# Visualisation of the impact of COVID-19 on air pollution

Krijn Doekemeijer[*]
Vrije Universiteit Amsterdam
k.doekemeijer@student.vu.nl

Tim Pelle[†]
Vrije Universiteit Amsterdam
t.pelle@student.vu.nl

Lucas de Geus[‡]
Vrije Universiteit Amsterdam
lj.de.geus@student.vu.nl

## ABSTRACT

The ongoing COVID-19 outbreak expanded rapidly throughout the world, forcing countries to take measures. The pandemic affected many people in different ways globally. Economies were slowed down significantly, resulting in a decrease of polluting gases. The emission of these gases can be measured in different ways and the TROPOMI installed on the Sentinel-5P satellite is one of them. To make this data more accessible to the public a visualisation of the measurements was created showing the evolution of the emission of the polluting gases before and during the pandemic. This was done by analyzing the data captured by TROPOMI, extracting the useful data in an efficient way to ensure an interactive and responsive website could be created for the visualisation.

**PVLDB Reference Format:**
. Final report: Group 08
Visualisation of Covid-19's impact on greenhouse gas emission. *PVLDB*, (): xxxx-yyyy, .
DOI:

## 1. INTRODUCTION

In 2020, the COVID-19 pandemic caused a lot of industries to temporarily close their doors and forced people to work from home. Many people were forced to quarantine themselves[1]. An example study by Harrington et al. [2] shows that these measurements caused the amount of traffic in the UK to decrease severely. Data published by TomTom[1] shows that the large cities Madrid, Rome, Paris and Milan have seen a sharp reduction ( >80%) in march of traffic volumes compared to the baseline of January 20th.
Open industries and traffic contribute to the emission of greenhouse gases, which would suggest that during the COVID-19 pandemic less gases have been emitted. Furthermore, the emission of greenhouse gases is a topic that is often a source of debate and is often linked to negative effects on the climate, which resulted in numerous research projects on pollution[3][4]. The impact of lockdown-measurements has impacted the air quality in western Europe [5]. In particular a reduction in $NO_2$, a smaller reduction in particulate matter concentrations and a mitigated effect on ozone concentrations were measured. Other studies show that the global effects have been similar [6]. However, most of these reports only consider a part of the bigger picture and do not show how the air pollution fluctuated on a small time scale, for example on a weekly basis.
The data for gaining such knowledge is already available and accessible. Much data is available which could lead to interesting research. However, actually accessing it and understanding the underlying datastructures is not an easy task. Additionally, translating these structures into concrete and understandable examples is difficult and requires a fair amount of effort and technical knowledge, which could make it hard for people to see the value of this data. To lower the threshold to this data and showing what is possible, this project aims to visualize the decrease in air pollution for the period in which the COVID-19 pandemic became more widespread and lockdowns were enforced. It will show that interesting results can be achieved with this data and will hopefully open the doors to more research and awareness using the large amount of data that is available.
In the next section relevant research, techniques and data will be presented that contribute to a better understanding of the themes related to this project. In section 3 the research questions for the project will be presented. These question will guide the research and ensure that the scope of the project is clear. Section 4 describes the execution of the complete project from the initial data investigation to visualizing the final product. Section 5 shows the results found during the project. Section 6 summarizes what has been done and explains the final results by answering the aforementioned research questions. Lastly, in section 7 the work is discussed and future work is suggested.

## 2. RELATED WORK

The related works have been divided into four categories: Data gathering, Pollution, Visualisation of pollution and Visualisation tools. Each category is discussed separately below.

[*]
[†]
[‡]

[1]https://www.tomtom.com/blog/moving-world/covid-19-traffic/


*Proceedings of the VLDB Endowment,* Vol. , No.
ISSN 2150-8097.
DOI:

## 2.1 Data gathering

To gain a better understanding of the presence of greenhouse gases in the atmosphere, various methods have been deployed to measure this. One of the methods that is used, namely measuring by satellite, will be used during this project. The Sentinel-5P satellite, developed by the ESA, uses the TROPOMI (TROPOspheric Monitoring Instrument) to measure the wavelength bands between the ultraviolet and shortwave infrared [7]. It uses passive remote sensing techniques to measure the solar radiation reflected by and radiated from the earth. By doing so, 7 particles and gases that are present in the atmosphere were measured, namely Methane ($CH_4$), Carbon Monoxide (CO), Formaldehyde (HCHO), Nitrogen Dioxide ($NO_2$), Ozone ($O_3$), Sulphur Dioxide ($SO_2$) and Aerosol. For each substance, a great amount of data has been captured by the TROPOMI. Some other works that describe how the data can be processed and visualised is discussed by for example Bauwens et al. [8]. Something else that was also discussed by many works is the accuracy of the TROPOMI dataset. One example from Wang et al. [9] discusses that there the TROPOMI dataset is still off on various occasions.

## 2.2 Pollution

The effect of COVID-19 on air pollution is something which has already been researched extensively. In the introduction it was already mentioned that a reduction in $NO_2$, a lower reduction in particulate matter concentrations and a mitigated effect on Ozone concentrations has been measured [6]. Bauwens et al. also investigated the impact of COVID-19 on $NO_2$ pollution using the TROPOMI and found that it significantly decreased, especially in Chinese cities with amounts up to -40% relative to the same period in 2019 [8]. Within this research only TROPOMI data was used for which the qa(quality assurance)-value is $> 0.5$ and the cloud fraction is $< 40\%$. To expand on this, a study conducted by Sannigrahi et al. found a reduction of $NO_2$ and CO over large cities worldwide during the COVID-19 period, indicating that a reduction in emissions is to be expected [10]. Menut et al. measured the impact on air quality over western Europe due to COVID-19 lockdown measures. Instead of using satellite data, the WRF=model [11] and CHIMERE models [12] were used to simulate hourly concentrations of a variety of pollutants over Western Europe. The research is unique because it takes meteorological conditions into account. The results again show a reduction in $NO_2$ concentrations ranging from -30% to -50% in western Europe and an increase in Ozone concentrations in urban areas [5]. In Wuhan it was shown that there there has also been a significant drop of 71% in $SO_2$ levels, a drop of 11% in HCHO and a decrease of 4% in CO [13]. However, the same research showed an increase in $SO_2$ in other places, for example Seoul and Tokyo. Having this information makes it relatively safe to assume that a fluctuation in $NO_2$ and $SO_2$ will also be visualized in this project.

## 2.3 Visualisation of pollution

There have been projects that focus on visualizing this air pollution. An example can be found at NASA [14]. There it is possible to compare the amount of $NO_2$ from this year with previous years. Another example can be found at IQAirMap[2] and IQAirEarth [3], which present the current situation on air pollution. This can be used as a guideline for visualising air pollution, but cannot be used for information older than today since it is live, so is more useful on how to properly represent a pollution map.

## 2.4 Visualisation tools

To publish or show the results in a web-based front-end Wang et al. argue that after mapping and refactoring the data into a correct format like GeoJSON or KML, multiple JavaScript libraries are available to visualize the created data[15]. Moreover, visualizing geospatial data is happening constantly around us in services such as Google Maps or Buienradar, and has therefore been the subject of various researches [16] [17] [18]. To visualize this data, specific tools and frameworks exist that apply to various use-cases. Examples of such use-cases are showing geophylogenies[19] or showing static datasets interactively[20]. For demonstrating GeoJSON files MapBox[4] offers a JavaScript library specifically for doing this.
From looking at these examples it can be concluded that tools can be found to aid in visualizing the data related to this project. This means that the focus can be shifted from front-end development to visualizing the data in a meaningful manner.

# 3. RESEARCH QUESTION

This research aims to contribute to the fourth objective set by TROPOMI: *To develop and improve air quality model processes and data assimilation in support of operational services including air quality forecasting and protocol monitoring.* Specifically, focusing on the change of air quality due to the COVID-19 pandemic and thereby contributing to the possibility of protocol monitoring. In accordance with this goal the following research question has been stated: ***How can the reduction of global gas emissions during the COVID-19 pandemic be visualized concisely and interactively?*** In order to be able to answer this question sufficiently, it has been split up into sub questions that focus on more specific problems:

- Sub-question 1: How can greenhouse gas emissions be visualised efficiently over time?
- Sub-question 2: How can COVID-19 events be summarized and visualised globally?
- Sub-question 3: How can large datasets containing geographical information over time be efficiently displayed to end-users?

# 4. PROJECT SETUP

This section will describe the development and execution of the project. It starts with the raw input data and the initial investigation on this data. From this data a scope was inferred. After this, the pipeline to process this data was created and it will be discussed how this pipeline was used on each substance. Then there is a short section about the way the COVID-19 data is read, transformed and used. Lastly, the visualisation of the written data is described.

[2]https://www.iqair.com/world-air-quality
[3]https://www.iqair.com/earth
[4]https://docs.mapbox.com/mapbox-gl-js/api/

```
In [29]: from netCDF4 import Dataset
         import numpy as np

         my_example_nc_file = '/Study/Master CS/Large Scale Data Engineering/S5P_OFFL_L2__CH4____20200607T000246_20200607T014416_
         fh = Dataset(my_example_nc_file, mode='r')

         fh.groups

Out[29]: OrderedDict([('PRODUCT', <class 'netCDF4._netCDF4.Group'>
                      group /PRODUCT:
                          dimensions(sizes): scanline(4172), ground_pixel(215), corner(4), time(1), layer(12), level(13)
                          variables(dimensions): int32 scanline(scanline), int32 ground_pixel(ground_pixel), int32 time(tim
         e), int32 corner(corner), int32 layer(layer), int32 level(level), int32 delta_time(time,scanline), <class 'str'> time
         _utc(time,scanline), uint8 qa_value(time,scanline,ground_pixel), float32 latitude(time,scanline,ground_pixel), float3
         2 longitude(time,scanline,ground_pixel), float32 methane_mixing_ratio(time,scanline,ground_pixel), float32 methane_mi
         xing_ratio_precision(time,scanline,ground_pixel), float32 methane_mixing_ratio_bias_corrected(time,scanline,ground_pi
         xel)
                          groups: SUPPORT_DATA),
                     ('METADATA', <class 'netCDF4._netCDF4.Group'>
                      group /METADATA:
                          dimensions(sizes):
                          variables(dimensions):
                          groups: QA_STATISTICS, ALGORITHM_SETTINGS, GRANULE_DESCRIPTION, ISO_METADATA, EOP_METADATA, ESA_MET
         ADATA)])
```

**Figure 1: NetCDF-file extracted**

## 4.1 Raw Input Data

All of the data that is used for gases comes from the Sentinel-S5 dataset. The Sentinel-5P dataset is represented in two different data formats. The first data format uses so-called COG(Cloud Optimized GeoTIFF)-files, which is similar to a regular GeoTIFF file except that it is being hosted on a HTTP file server and is said to be more efficient for the cloud [21]. A GeoTIFF is a TIFF file that holds images and georeferencing information[5]. This can be anything from coordinate systems to dates. An TIFF image also holds potential "bands", which are extra layers that hold additional information.
The other file format is called NetCDF, which is a is a format that can be used for array-oriented scientific data [22]. NetCDF-files do not contain images, but hold an incredible amount of data related to the TROPOMI measurements. For measuring the presence of a certain gas the TROPOMI not only stores coordinates and presence and volume, but for example also accuracy variables, quality related variables, wind speed and much more. For each substance detailed documentation is also available. A summary and "easy read" for for example $NO_2$ is available under [23]. Since detailed documentation for each substance is available, it became easier to decide what variables were needed for analysis as well as finding the right scales and thresholds. Through this research it was found that generally four arrays of data are of interest, namely the longitude, latitude, data and qa(Quality Assurance)-value. The first three variables are clear, they represent per coordinate the measured presence of the gas in question. The qa-value represents for each "pixel" in the image how clean, or better, uncontaminated the measurement is by factors such as reflection or clouds. Generally, values below 0.75 should be dropped, but this is dependent on the substance. Unfortunately, the NetCDF format also has a downside which is that files can be very large and cumbersome to extract portions of data from. Depending on the substance, file sizes can range from around 50MB in the case of $CH_4$ up to around 750MB or more in the case of $SO_2$.

[5]https://www.earthdatascience.org/courses/use-data-open-source-python/intro-raster-data-python/fundamentals-raster-data/intro-to-the-geotiff-file-format/

In the end the NetCDF file format was chosen because it contains more information than the GeoTIFF files and in case this information was needed it would be quite cumbersome to move to NetCDF. Further on using NetCDF allows for more control over what data will be used and displayed. Each of the formats contains 3 data streams. These are the near real-time, offline and reprocessing data streams. The reprocessing stream only contains data from 2018, so this is not very useful for the scope of this project. Also, the near real-time stream only contains data from March 2020 until June 2020. However, the offline processing stream contains data from various years up to and including June 2020. To ensure that the data for visualisation is reliable and since the aim of this project is to compare data from various months and years, it was decided that it is necessary to get data from a single stream. Hence, the decision to use the offline processing stream. This stream contains data from each day of each month of the year 2018 to June 2020 for most of the substances under investigation in this project. Furthermore, it was found that between the offline and near real-time streams no real difference in data existed, therefore no important data would be missed in choosing only the offline stream. The daily data that the TROPOMI generates for this stream does not come from a single measurement. Each day the satellite usually creates between 12-14, and sometimes more, pieces of data. If added together these pieces create an approximated view of the entire earth. However, the data may still be incomplete because some parts were not scanned or due to external factors. The total size of the offline data stream is 20 tebibytes.

## 4.2 Initial Data Investigation and visualisation

Now that the decision was made for which data to use, some initial data investigation and visualisations were made. This gave a general idea of what would become the end product. During the start of the project the main goal was to understand the data at hand. As has been explained in section 4.1, two data formats were available. At first, the obvious choice was to use the preprocessed COG files, since they contained the images already. However, this would limit the ability to design a visualization that would fit the purpose of this project the best. Of course, the NetCDF files were analyzed as well. The NetCDF from 17 Oktober

2018 was extracted by using NetCDF4[6] for python. The data that was found is shown in figure 1. In this figure one can see that the file is split into groups, which have dimensions and variables. Also, high-level groups have subgroups, which may have more data attached to them.
To understand this format better and to get an idea of how the visualized data might look like, an application called Panoply was used[7]. This application has built-in support for opening NetCDF files and displaying these on a map. Figure 3 shows the qa-values of a certain NetCDF-file while Figure 4 shows the measured Carbon Monoxide (only if the qa-value is >0.5 is the measurement taken into account)
After the initial data investigation, a basic visualisation using the NetCDF files in python was realised. Figure 2 shows a single measurement and visualises naively the tropospheric vertical column values of $NO_2$ over a set of longitudes and latitudes. This particular figure is a single file from October 17th, 2018 and is therefore only a part of the data of this day. All "pixels" that have a qa-value below 0.75 have been dropped, as has been explained in section 4.1. Furthermore, to compress the data, all latitudes and longitudes where averaged to 1x1(1 in longitude and 1 in latitude) squares, which meant that multiple measurements have been combined to a single measurement. We exported these squares to a GeoJSON [24] file with the help of GeoJSON for Python[8]. We then loaded this file back and displayed it using python. This result was then compared with the Panoply example and the two were similar, which ensured that visualising the data in a meaningful way is definitely possible.

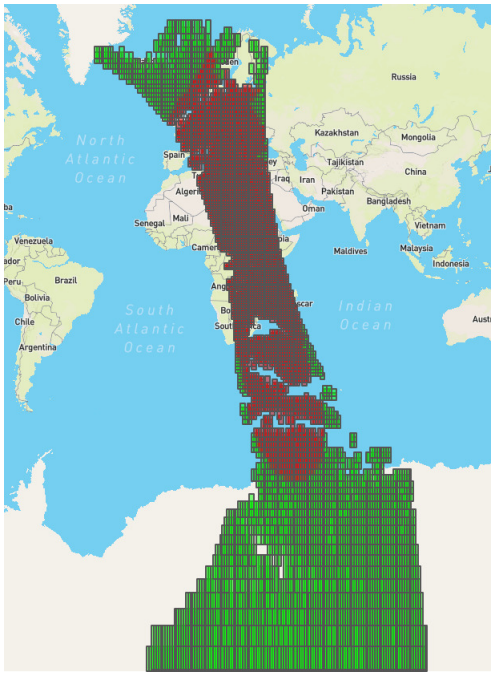


**Figure 2: Single Measurement of NO2**

[6]https://unidata.github.io/netcdf4-python/netCDF4/
[7]https://www.giss.nasa.gov/tools/panoply/
[8]https://pypi.org/project/geojson/

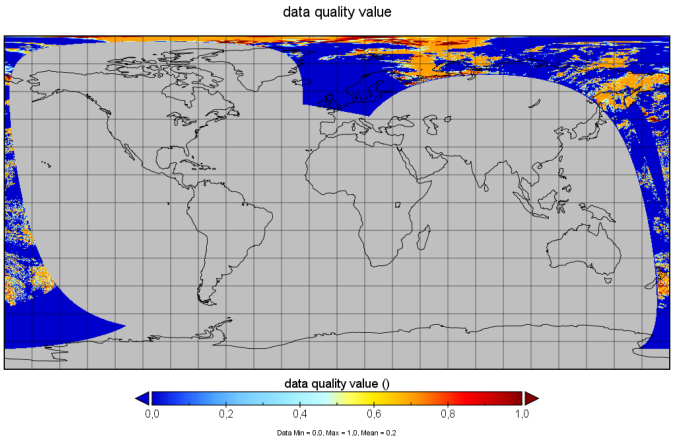


**Figure 3: NetCDF-file qa-values**

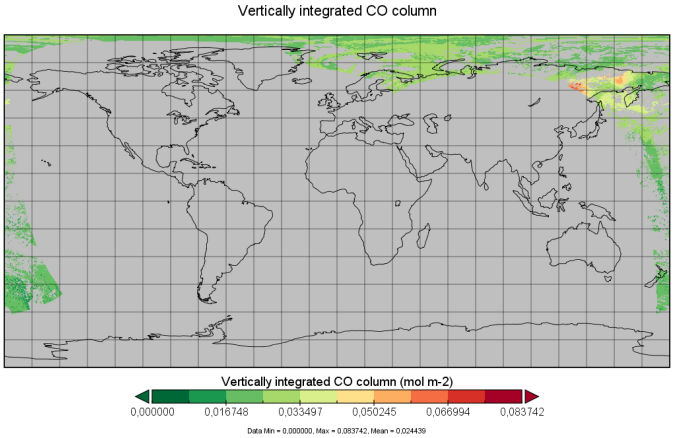


**Figure 4: NetCDF-file carbon monoxide values**

## 4.3 Scope

As was explained in section 4.1, only the offline data streams for NetCDF are used. Since the offline datastream ends on the 14th of June 2020 and one of the goals of the project is to visualize the difference between emissions before and during COVID-19, it was decided to only pick data between the 1st of January and 14th of June. This means that for both 2019 and 2020 only the data between January 1st and June 14th are picked. Furthermore, it was decided to only pick polluting gases that have been found to have changed during the pandemic in quantity and that are the most harmful to the environment. Because of scoping issues, it was also necessary to only look at a few substances. In the end 4 substances were picked: $NO_2$, $SO_2$, CH4 and CO. $NO_2$ was chosen because this substance was proven to have decreased throughout the pandemic [6]. $SO_2$ was chosen because according to some sources it changed in a few countries, both positively and negatively [13]. $CH_4$ was chosen due to sources stating to have decreased during the pandemic (Talekdar et al [25]). A decrease in this substance would be important, since this gas is said to be a more potent greenhouse gas than $CO_2$ [26]. Lastly, CO has also been found to have decreased during in the pandemic in mayor

cities worldwide [10]. Ozone would be a good candidate for the visualisation as well since it is said that the negative effects on the ozone layer have been mitigated [6]. However, it is a lot less country specific (mainly visible around the poles) and a visualisation might therefore be less interesting to look at than the other candidates. HCHO was not picked because it is not a greenhouse gas and also does not harm the ozone layer. When writing files it was also noticed that writing data for each day and each coordinate was a lot of information. One of the first things to reduce the size of the dataset was averaging all coordinates to larger zones. These zones all range .1 degrees in both longitude and latitude directions. Since longitudes and latitudes are not equal everywhere on the earth, it leads to some zones that are a bit more stretched. This will be discussed more in depth later on. Additionally, it was decided that rather than showing the results for each day, to show it per week. As a result the data is averaged over the zones for all snapshots of a day as well as all days in a week. Some further data reduction was done by limiting the scope only to data on actual land. The reason for this is that the main interest of this project, as can be read in section 3, is to visualize the change in emission of certain gases during COVID-19. The emission of these gases mostly happen on land and also the lockdowns were enforced mostly on land, thus making the reduction most visible on those coordinates.

## 4.4 The pipeline

The entire algorithm for the pipeline was first tested on local machines, before it was parallelized on Databricks. Therefore, the algorithm that is used to extract-, modify- and write data is written to be easily modifiable and parallelizable and should function on local machines and with Spark. The algorithm itself is divided in 4 distinct sections: reading and filtering data, averaging and combining data into tiles, creating JSON files and writing the JSON files. Each of these sections will be discussed. In addition to the algorithm a few global variables that need to be set for each substance are kept. This makes it easy to run the program for different substances with minimal change needed.

### *4.4.1 Data extraction*

All of the data extraction can be traced back to a Python 3 function that reads NetCDF files. This function is ignorant of the underlying TROPOMI directory structure and only reads an absolute path to a NetCDF file. It is also by far the most expensive function in the program both in time and processing power. It then uses the Python library "NetCDF" to parse the NetCDF file to a structure that is easy to read and modify. If this function throws an exception, the program crashes. This means that this function does not allow for corrupt records. This is handled elsewhere. When the data is parsed, only a few arrays need to be kept. The NetCDF structure mainly contains arrays in a structure that is not too far away from a columnar data store. However, an important thing to note is that almost all arrays are stored as 2D arrays. These are a latitude, a longitude, a quality value and a value array. Only the value array differs between substances and needs to be handled differently. All of the other arrays in the NetCDF file are simply dropped. After this, the data is filtered to only the row entries that are needed. This was first done in an order that was not optimized and proofed to be fault intolerant. It also used many explicit for loops that did not make full usage of the underlying hardware. In order to achieve a reasonable speed, it was decided to make use of mostly Numpy functions, which are known to be easily parallelizable (talking about local parallelliziation here) and make use of optimized C functions instead of pure Python functions [27].
It was noted that all of the necessary arrays have a similar 2D shape. That is to say, they their width is equal, their height is equal and they are all 2D. This means that if we keep track of what rows we need, we only need to keep track of one 2D mask array that contains for each element if we need it. This mask can then be applied to all columns that are being kept. The first filter used in the mask is to remove all rows that contain corrupt latitudes and longitudes. This is done as early as possible because later functions assume that all latitudes and longitudes are valid, since doing constant checks for bounds is expensive). Then the second filter is applied to the mask. Each entry has a quality value (qa-value) as was already mentioned in section 4.2. Only the data that is above a certain threshold is usable. This threshold can be tweaked for each substance and everything below this threshold is dropped. The third and final check only works with valid latitudes and longitudes. Therefore the mask is first applied and then this check is done. This check is also expensive, which further justifies doing this filter as late as possible. This filter checks if the data is on dry land.
After all data is filtered, it is combined into something that can be used later on. In order to reduce the amount of data for both computation and storage, only a portion is kept. For this the data will be averaged to .1 latitude and longitude accuracy as was explained in section 4.3. This averaging could however make the image look a bit more distorted especially along the poles. That is because that is because longitudes and latitudes differ along the Earth. Near the poles longitudes may reach close to 0 km distance between each other. This decisions was made because it is not trivial and expensive to go from latitude and longitude to kilometers and back. Further on, the underlying map is distorted itself as well, which should also be accounted for. This can be seen because the visualisation is on a rectangle map, whereas the earth is a sphere. In the end this risk was migitated and a decrease in accuracy along the poles was accepted. The averaging itself is postponed, but all data belonging to the region that will be averaged, will be kept together. For this a dictionary is used. In order to make sure that not too much memory is used and information is cheap to convert back and forth an int is used as a key. Latitudes range between -180 and 180 and longitudes between 90 and -90. This information can be used. It is possible to go to .1 accuracy by multiplying with 10 and then rounding back to an int, for example 5.34 should become 53 and -6.67 should be -67. For later usages, it is necessary to make all numbers unsigned, therefore longitudes get an additional 1800 and latitudes an additional 900. So longitudes range between 0 and 3600 and latitudes between 0 and 1800. These numbers can then be combined with the formula:

$$Key = longitude + latitude * 3600$$

This is similar to how 2D arrays can be encoded into a 1D array and is similar to the formula $index = x + y * width$. Do note that it is absolutely necessary that all values between 0 and 3600 belong to latitudes and only 3600*x belongs to

longitudes. The longitude can later be retrieved with the formula

$$longitude = \frac{key\%3600 - 1800}{10}$$

and the latitude with

$$latitude = \frac{\frac{key}{3600} - 900}{10}$$

. The value in the keyvalue pair will be a tuple of (totalValue, amountOfValues). Each value that will be added to a key adds its value to totalValue and 1 to amountOfValues. These tuples can later on be reduced to an average with $\frac{totalValue}{amountOfValues}$. This is not too different from a combine-reduce solution like would be used in MapReduce[9]. It is possible for external functions to give an initial dictionary, which would allow multiple days or even weeks to make use of the same dictionary.

### 4.4.2 Averaging and combining data into tiles

Each task is assigned an x number of so-called "day" tasks. This is usually a week with each day containing multiple files, see section 4.1. Each day therefore represents a directory. This directory is scanned for files and only the NetCDF files are selected (other filetypes might also exist). The data extraction discussed in section 4.4.1 is performed on these files. As was noted in section 4.4.1 corrupt files will crash the extraction. Therefore, some error handling is done while running the data extraction function. This is done with a try-catch function. This is not the most elegant, but accounts for almost all possible errors. A myriad of errors can happen when reading a lot of files and doing all tasks over is expensive. When something goes wrong, the task should proceed, but log the errors. If an error occurs, no data is added that was read during the extraction. Rather, the error is logged to an error log file. This can be used later on to verify what went wrong and how often this occurred. After all data is read, some averaging will need to be done. For this a dictionary is used. As was explained in section 4.4.1, the data extraction method accepts a dictionary. This allows the task to write all data to the same dictionary. This dictionary is then passed on to a different function. This function goes over each key value pair and applies the conversion from key to (latitude, longitude) and calculates the average over the value as has been explained before. All of these results are stored in an array of tuples of (latitude, longitude, average). The visualisation on the website requires tiling for more efficient rendering, see section 4.8.2. Therefore the entire world is divided into twelve regions. Each region has the complete latitude, but only 30 ($\frac{360}{12}$) in the longitude region. To account for this, 12 arrays are created. The tuple is added to the array with index $\frac{longitude+180}{30}$. This formula ensures that all data is written to the proper tile-array.

### 4.4.3 Creating JSON

In this project, the GeoJSON format is used for storing the data that will be visualised. This file format can be quite verbose and also requires a lot of repetition. The reason why GeoJSON is used, is that the MapBox Javascript API requires a standard format such as GeoJSON. It should in theory be possible to write a simple format and write a "convertToGeoJSON" function in Javascript that does this on the fly. This was not done because the GeoJSON format can take significant time to load into MapBox and switching from one dataset to another would add to this waiting time and therefore give a decrease in responsiveness. In order to reduce the disadvantages of GeoJSON, some workarounds were needed. Firstly, it was determined what was absolutely necessary to be stored. For all of the (latitude, longitude, average) variables, the latitude and longitude were absolutely needed. That is because GeoJSON requires these as pairs and due to the averaging all of these values are unique. However, the value that accompanies the coordinate pair is not necessarily unique. In fact, it can be observed that many locations have a very similar average. In a later stage, this averaged value will be used to show a color on the map. Also, it is not necessary to encode thousands of different values, if they only end up being used for colors. The user will probably not notice 1000th of an increase in red. Instead only a few hundred are necessary, which was incorporated into a design decision. So, only a few hundred values are allowed, but the exact numbers differ slightly per substance, but all points with similar values will be grouped together. The points can be grouped together by adding them to a so-called Multipoint[10]. Now only one a few features need to encode the value. However, saving the value causes a different problem, namely the data had to be interpreted to ensure that it is usable for the front-end. In the end, it was decided to convert the value to a color when creating the GeoJSON. Not only is this better for the performance for the site, it also ensures that the site does not have to know anything about the actual values. The actual conversion to a color is explained in section 4.8.3, but to explain it shortly, all values are converted to a hexstring of "#RRGGBB" with R representing the red bits, G the green bits and B the blue bits. The end result is a GeoJSON of the format:

```
{
    "type": "FeatureCollection", "features":
    [
        {
            "type": "Feature",
            "geometry": {"type": "MultiPoint",
            "coordinates": [[lon, lat], ...]},
            "properties": {"fill": "#RRGGBB"}

        },
        ...
    ]
}
```

The "FeatureCollection" can be seen as the root node with a list of "MultiPoint" nodes as children. Each MultiPoint contains an array of longitude/latitude pairs and a unique color defined as properties. All of the tiles discussed in the previous section, will be converted to this GeoJSON format individually. Later on, it became evident that certain rows could be corrupt. These rows are not written to the JSON, but are instead written to a dedicated log file. This makes it easy to see how much data is lost and especially what is lost.

[9] https://event.cwi.nl/lsde/papers/mapreduce-osdi04.pdf

[10] https://tools.ietf.org/html/rfc7946

#### 4.4.4 *Writing the GeoJSON*

After all GeoJSON formats are created, they are written to files. This function is made explicitly simple. It only accepts JSON and a filename and other tasks have to give the data and a filename. This allows it to be easily used on both local systems and with Spark on Databricks. There is however also a more general function for writing tiles. This function not only recognizes tiles in data, but also creates a JSON for each tile and writes it. It recognizes a file prefix and for each tile it appends the tilenumber and ".geojson" behind it. Tiles should in general be written from left to right. This means that in the data array the first tile should start at -180 degrees as the longitude and the last should end at 180 desgrees. It also means that the first tile will be written as prefix1.geojson and the last as prefix12.geojson.

## 4.5 Running the algorithm on Databricks with Spark

The algorithm described in the previous sections allows for easy porting to the Databricks environment. To make the best use of the parallelization it had to be decided what tasks are applicable for this. The tasks had to be individual and had to have a good granularity. It was noted that each substance clearly needed different arguments and also strictly contains different data. In order to properly test visualisations on the webpage and get a good workflow, only one substance per year is written at a time. The reason is again mainly for easy verification and testing. Running both years, 2020 and 2019, would take a significantly larger time to run. The tasks are big enough to require little user input, but are small enough to be testable. Therefore, for each substance, for each year a separate "notebook" is used even though they were run in sequence. Each "supertask" would now exist out of 26 weeks, see section 4.3. Each of these weeks can be read completely independently and will be written to a different file. These can therefore be defined as separate tasks. Each task will get a week of input filepaths and one output prefix. The input filepaths will be paths to the S3 bucket. The output location will be structured like ".../Year/Substance/x_y" with x representing the weeknumber and y representing the tile number. Creating the tasks by hand could take quite some time, so a separate task can be run to create an array of tasks for Spark in "taskcreator.py" called "createTasks". These values need to be copied to Spark, because this will not run on Spark where the Python version is not at least 3.8. For clarification, the pipeline has been visualized as is illustrated in figure 5. It performs a task for each substance/year combination, then assigns a week for each worker and lastly runs the worker. All of these tasks are independent and are visualised in figure 6. The tasks starts by processing all days of the assigned week. The "Day task" and "Process NC file" are explained in section 4.4.1. Notice that if something goes wrong during processing the NetCDF-file it is logged. After this is completed the data is averaged and tiled. For each of the tiles a JSON is created and written. Again errors during the creation of GeoJSON are logged.

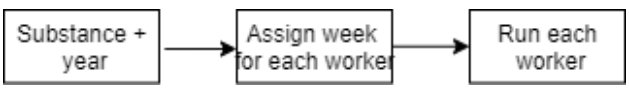


**Figure 5: The pipeline for all tasks**

## 4.6 Handling each substance

Each of the substances should be handled differently. This is because each of them have not only a different distribution, but often also a different measurement unit. For each substance a different colormap was used to ensure that differences between regions were easily visible. Not all data could be represented accurately, therefore some decisions for each visualisations were made, which are explained in the next sections. In general the visualisations are made with a colormap. A colormap represents a function that returns a color from a value based on the minimum value and maximum value. The minimum value and maximum value are used for interpolation. For example, assuming it is desired to show low values as red and high values as green. This requires a function that creates colors in between red and green, based on the values. Therefore, for each substance a minimum and maximum value had to be chosen. Even if these were not the true minimum and maximum values.

#### 4.6.1 $NO_2$

The NetCDF files of $NO_2$ contain data for multiple atmospheric layers. The set of $NO_2$ holds information about the tropospheric, the stratospheric and the combined (sum of the previous ones) atmospheric layers. Both the tropospheric and the stratospheric layer are visualized. It was chosen to show both separately since they might show different data. The sum was not chosen because it would be harder to deduct conclusions about the emissions when the two layers are combined. However, it was decided to use the same colormap, minimum value and maximum value for both atmospheric layers to ensure that a comparison between the layers would be clear and easy to see. For the tropospheric layer the array "nitrogendioxide_tropospheric_column" was used and for the stratospheric layer the array "nitrogendioxide_stratospheric_column" was used. For both layers the unit of the measurements are in mol/$m^2$. Unfortunately all $NO_2$ values are really small, therefore a better measurement unit for the visualisation would be $\mu$mol/$m^2$. The reason for this is that the values then can be represented as full integers. It should then also be possible to create a mapping from value to color. However, for this a colormap is needed. It was noticed that there were some values below 0 and a few above 150. However, most values were in between 0 and 100. If a really large set of values was chosen, say -100 to 500. It would mean that a lot of the colorspace would be wasted on very few values. This would mean that most of the visualisation will be limited to one color. Therefore the minimum and maximum values were chosen, to represent most of the data, but still visualize possible outliers. For this other visualisations of $NO_2$ were examined. For example Bauwes et al [8], Duncan et al [28] and the visualisation on the actual Tropomi data products site[29]. Many of the visualisations use a colormap that is similar to the Matplotlib colormap "jet", with some slight changes like using green instead of white. This map represents a gradual increase from dark blue to light blue to green to yellow to orange to red. This is quite a wide spectrum and makes it easy to spot small differences between regions. There was however not a lot of consensus on min and max values. In the end it was decided to use 0 and 110 $\mu$mol/$m^2$ based on trial and error. Values below 0 were simply converted to 0, because they were seen as at least as "positive" as 0. The value 110 was chosen because any value that is lower, does

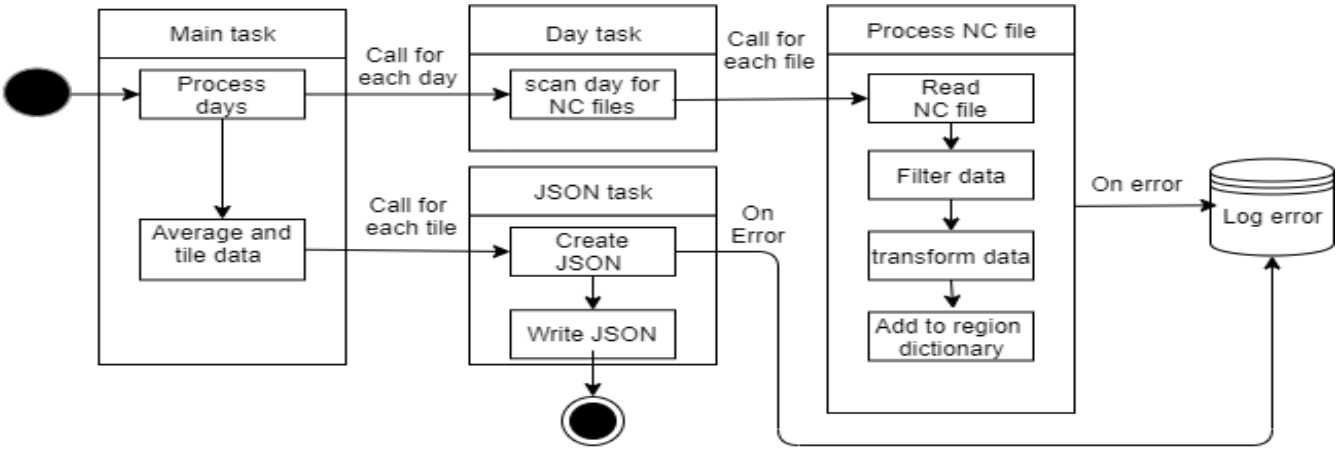


**Figure 6: The pipeline for one task**

not correctly show outliers and any value above makes it hard to see differences on the map. Any value above 110 is simply treated as being 110.

#### 4.6.2 $SO_2$

For $SO_2$ there is only the combined column of tropospheric and stratospheric layers. So it is only possible to visualize the sum of both layers together. This makes it harder to pinpoint if there is a lot of $SO_2$ in lower or upper layers of the atmosphere. The data is stored in $mol/m^2$ just as with $NO_2$. For $SO_2$ the visualisations from NASA [30][31] and Tropomi[29] were examined. Those visualisations all use so called "DU" (Dobson units)[11]. A few visualisations used 2 DU as a max value, others used 1. This was tried as well. For this 2 DU was converted to $mol/m^2$, this can be done with the formula $1DU = 0.4462mmol/m^2$. It was noticed that there were a lot of different values next to each other. This resulted in lots of different colors next to each other when using the "jet" colormap. This is visible in figure 7. To tackle this problem a colormap was used that used less colors and a more gradual change. This color map was called "RdYlGn_r" and is again from Matplotlib. It represents a shift from green to red. Unfortunately, there was now mainly one color visible on the screen, green. Apparently, a lot of values are below 2 DU. With some trial and error, the end result became 0 DU as a minimum and 0.5 as maximum. This particular scale allows the user to see differences on the map, but not too much. Here it is important to note that this means that many extreme outliers are not properly visible. The reason why "RdYlGn_r" was used is because of a few different reasons: the values fluctuate too much for a colormap like "jet", most colors are in a lower spectrum and lastly it avoids confusing the data with $NO_2$. The second reason "most colors are in a lower spectrum" requires some additional explanation. The human eye is more sensitive for green than for blue [32], since most values will fall in to the lower spectrum, green would therefore be ideal.

#### 4.6.3 xCH4

Methane or xCH4 should be handled quite differently. Rather than a certain quantity in a layer, parts per billion (ppb) are measured. This means how many particles in a billion are CH4. So the result is a mixing ratio of methane and other substances. This variable is stored as "methane_mixing_ratio". For this mixing ratio, TROPOMI already defined a minimum and maximum value that can be used for histograms [29]. These values can be found back in the product user manuals that are provided. The used minimum and maximum values for the histograms were noted as 1200 and 2000 respectively. This was tried and seemed to work well enough, but could do with some improvement. Therefore some other visualisations were analysed. These are by Frankenberg et al [33] and Turner et al [34]. These all provide ranges between about 1700 to about 1900. Unfortunately, methane emissions have increased over the years and these are older visualisations [26]. In the end a minimum of 1700 and a maximum of 2000 was used. After these values were found the approach to xCH4 was most simplistic one. That is no conversion is needed, since the data is given in ppb and displayed in ppb. There was however one caveat. Methane requires a different qa-value than the other substances, namely 0.8. This, in addition to containing less data generally, meant that a lot of data seemed to be missing. This resulted in a lot more gaps in the visualisation compared to the other substances. The colormap that was used in the end is the Matplotlib colormap "gist_heat_r". This colormap shows a good progression from low to higher values. Values that are near the minimum are white/yellow while values near the maximum are a deep red.

#### 4.6.4 CO

Carbonmonoxide or CO is measured in $mol/m^2$ just like $NO_2$. It should also be handled quite similar to $NO_2$. The column that should be retrieved is called "carbonmonoxide_total_column". Unlike all other substances, the threshold of qa-values can be set to 0.5. For visualisation the TROPOMI website was analyzed [29]. Unfortunately the measurement unit that is used is in ppb, not in $mol/m^2$. This conversion is not trivial and is not the purpose of this project, therefore it was decided to not convert to ppb. Unfortunately, no visualisation was found that made use of anything other than ppb. This meant that the scale had to be chosen by hand. By analyzing a few samples, it was noted that all values are far below 1 and converting to $mmol/m^2$ gave a better scale. In this scale integers could be used. The

[11] https://sacs.aeronomie.be/info/dobson.php

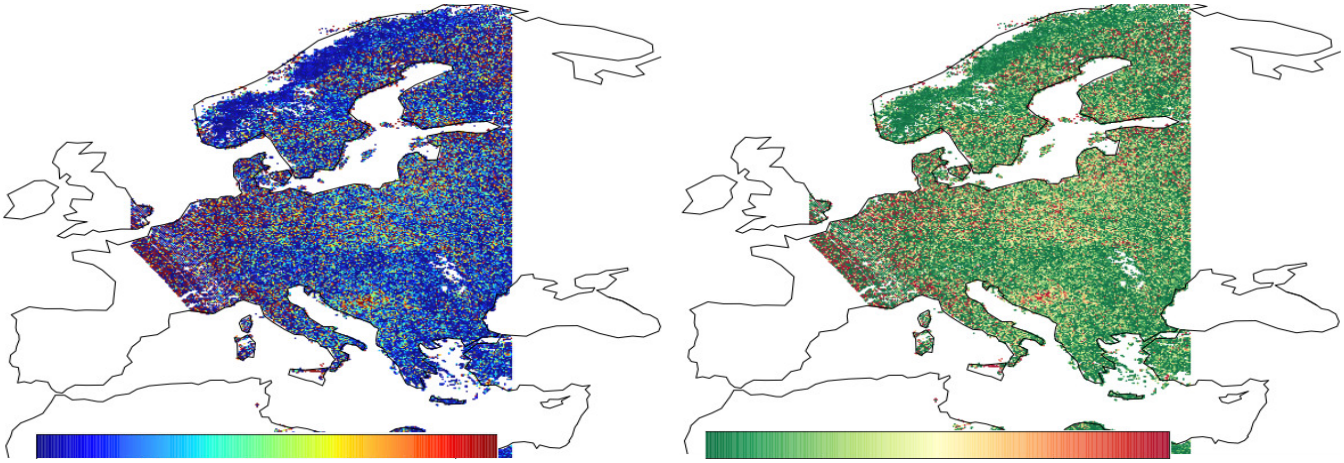

**Figure 7: Difference in SO2 left colormaps. Left: jet, Right: RdYlGn_r**

main tool to test visualisations ended up being Panoply. One of such tests is visualized in figure 8. As will be discussed in section 7, basing the visualisations on a few samples is quite problematic. In the end, the Matplotlib colormap "coolwarm" was used. This colormap makes it clear to the user that they see a different substance than before, since each substance has its own colormap, and gives a clear distinction between low and high values. Low values are represented as blue and high values as red. Values in between the middle are white. The values that were used in the end were 0 and 140. Even though many significantly higher values were measured, it is clear that many values are really close to the minimal (dark blue), therefore to allow for some more contrast a lower maximum value was chosen. Do note, that unlike the other substances, this is less empirical and might have not been the best range for each week.

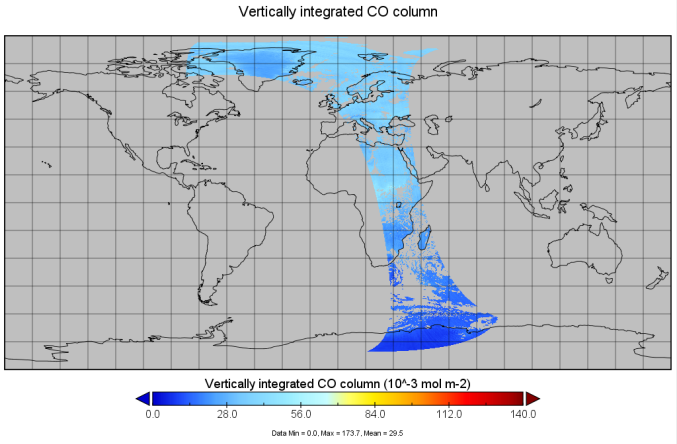


**Figure 8: Carbonmonoxide in Panoply**

## 4.7 Covid-19 data

To visualize the COVID-19 pandemic it was decided to use the strictness of 'lockdown styles' which countries have had during the pandemic as a representation. This strictness is a number that ranges from 0 to 100, extracted from a dataset[12] provided by the University of Oxford and Blavatnik School of Government. The number is based on 17 indicators that fall under three categories: closure policies, economic policies and health system policies. The dataset contains the daily strictness numbers for each country since the 1st of January, 2020. However, it might be the case that the strictness number is significantly higher or lower in certain regions of a country. For example, some cities might have a full lockdown while other cities do not. This gives some data inaccuracies, but it does give a general idea of what lockdown status countries were in on a given week. The second dataset used to visualise COVID-19 is provided by Google[13] and contains the centroids of each country, given by latitude and longitude. By merging the two datasets a GeoJSON file is created that contains the coordinates and strictness number of each country on different dates. This allows the visualisation of the global COVID-19 pandemic over time. The algorithm for merging the datasets and the file format are discussed below.

### *4.7.1 Algorithm*

The python library Pandas[14] was used to work with the datasets. Since the COVID-dataset contained dates in a format that Pandas could not read, the column containing dates was first formatted to the desired format. Then the datasets were joined on their mutual column: "Country names". However, some entries from the COVID-dataset contained strings in the "Country Name" column that did not exist in the centroid-dataset. This was due to the fact that some countries names were spelled differently in the datasets (e.g. Slovak Republic - Slovakia, Congo - Congo [Republic]). Therefore the centroid-dataset was manipulated to have the same spelling as the COVID-dataset. After the join, a dataframe containing the columns: date, longitude, latitude, strictness and countrycode remained. For each entry a GeoJSON point was created and added to a collection which was exported to a GeoJSON file. Additionally, each point was given a RGB-colorcode based on the

[12] https://www.bsg.ox.ac.uk/research/research-projects/coronavirus-government-response-tracker
[13] https://developers.google.com/public-data/docs/canonical/countries_csv
[14] https://pandas.pydata.org/about/

strictness number, allowing easier visualisation (low strictness: green, high strictness: red).

### 4.7.2 JSON

The JSON format for COVID-19 data is handled a bit differently. There is at most one element for each country, for each week in 2020. That means that there would only be a few thousand points in total. Separating this data in layers would therefore not be necessary. Furthermore, it should not be necessary to write each week to a separate file. Loading extra files is quite expensive because that would mean re-sorting all the other layers as well (layers in the order last in, last out). Therefore it was decided that all data is kept in a single GeoJSON file. In this GeoJSON file each entry is encoded with a week property. This week property can then be used to filter only the entries that are needed for each week. Rather than MultiPoints, Points are used. That is because the amount of data does not justify this performance treatment. In the end the following simple format was used:

```
{
    "type": "FeatureCollection",
    "features":
    [
        {
            "type": "Feature",
            "geometry":
            {
                {"type":"Point,
                 "coordinates":[lat,lon]
                },
                "properties":
                {
                    "name":"XXX:Y",
                    "week":int,
                    "fill": "#RRGGBB"
                }
            }
        },
        ...
    ]
}
```

The XXX is a placeholder for the countrycode like NLD for the Netherlands. The Y stands for an integer representing the weeknumber. The hexastring for color works the same as it does for the substance data.

## 4.8 Visualisation

While for each substance the data was extracted and processed, it could already be used to create a clear visualisation and start answering the research questions. For the purpose of this project a webpage using HTML, CSS and Javascript was created. Fortunately, certain libraries already exist that could be used to map the data on. After investigating possibilities of mapping the data on a map in a meaningful and interactive way two options were found which have already been mentioned in section 2.

### 4.8.1 Libraries for visualisation

At first Folium[15] seemed to be a viable option. It supports the GeoJSON datatype and ensures a certain flexibility in visualizing the data. The data could namely be created as so-called vector tiles, which take the coordinates, geometry features and colorscheme of each substance (see 4.6) of the GeoJSON as input and show them on a map. Unfortunately it would take a lot of effort to show the data results in a interactive manner, which is one of the main goals of this project.
Fortunately, a library already exists that builds upon the same basis, supports the right dataformats and has specific functions for showcasing the data interactively. Through MapBox it is possible to either upload the data to the service and create a unique visualisation through a GUI or use local files and create tiles and the mapping through a Javascript library. Using the GUI was found to be easy, but unfortunately not usable since the service became costly fairly quick once tiles of the whole world would have to be visualised. Instead of the GUI the Javascript library MapBox GL JS[16] was used.

### 4.8.2 Visualizing Emission Data

Mapbox GL JS provides many useful functions that allow to create interactive maps visualizing GeoJson files. The way this happens is that a basic empty map is created as a source on which layers in the form of vector tiles can be added. To ensure that rendering the website does not take too much time and computing power, GeoJson files were created that contain emission data for a week for a certain region of the world (see 4.4.4). These GeoJSON files are loaded as tiles by the MapBox GL JS library. To make sure the application is user-friendly we have written custom Javascript code that prevents all tiles being loaded in to the cache memory of the user at the same time.

By retrieving the user's location it is determined which regions should be visible. The corresponding tiles (see 4.4.2) for these regions can then be downloaded. We restrict the zoom-level of the map in a way that the user can view at most 5 tiles at the same time. Besides the 5 tiles that have to be downloaded for the current view of the user, we also download the same tiles for the previous and next week to allow smoother switching between the weeks. If tiles are at any point in time not needed anymore they are removed, once they become needed again they are downloaded again. By doing this, at most 15 tiles per map are stored in the users cache memory. With this approach we choose not to minimize data exchange but to minimize the data that must be present in the cache memory of the user.

Since the goal is to show the difference between emissions before and during the COVID-19 pandemic two basemaps were created and put next to each other. Each map shows data taken from the same week of the year for a certain substance, but from different years, namely 2019 and 2020. By putting these maps next to each other a fast and clear view on the differences is shown. To these two maps some functionality was added to ensure that the views remain clear and navigation is possible. Firstly, a time slider is added that allows the user to switch between weeks and shows the evolution of the emission per week. Secondly, buttons are added that allow the user to switch between different substances. Furthermore, the two maps have been synchronized in order for the user to compare the emission per region and not get lost or having to navigate through two maps to see the data of different regions.

[15] https://python-visualization.github.io/folium/

[16] https://docs.mapbox.com/mapbox-gl-js/api/

#### 4.8.3 *Visualizing Covid-19 data*

Visualizing COVID-19 data along with the substances proofed to be quite an undertaking. For each country a value for the lockdown should be displayed. Using simple shapes like the points used for substances, would not have worked well. That is because it would be impossible to see exact values. Therefore, it was decided to use text instead. Similar to the other data, the color should differ based on the value. However it is not trivial to choose these colors. They may not interfere with the underlying substance visualisation and the text should remain readable. The end result was drawing a black box on top of each centre of a country. No substance has a colormap with black, so it does not disturb the other visualisations. On the contrary, it adds some contrast. On top of this box the text was drawn in a color based on it's value. Lockdown numbers that were exactly 0 were marked as green. That is to make it easy for users to see that no COVID-19 measures are active in a country at that moment. If the value ranged from 1-25 it is drawn as orange-yellow. This makes it easy to see that some measured are taken, but it is not that heavy. If the value trespasses 25 it becomes more red. For each 25 integers, it gets one tint more red. These values were carefully chosen to still be readable and to be distinct enough to notice. Since the texts are not directly next to each other, it would not make sense to add really small increases. Therefore only big leaps were chosen. However, now a different problem came into existence. The lockdown measurements combined with the substance data, would obfuscate the country name. The solution for this was to add 3 letters for each country, the CountryCode. This is long enough to make countries recognizable, but not too long that it would obfuscate the substance data.

# 5. RESULTS

In this section the results that were obtained from the project plan are discussed.

## 5.1 Data reduction

The original dataset of 20TiB was significantly reduced, which was mostly done by focusing only on some sub-directories. However, by exposing these sub-directories to the pipeline described in 4.4 it was possible to decrease the size by **99,9**% on average. Table 1 shows the original directory sizes against the size of the data used for visualisation. For $NO_2$ we have visualized the troposphere and stratosphere, therefore they are mentioned separately to better visualize the data reduction. The final dataset that is used for the visualisation, including substance-data, COVID-19-data and colormaps (PNGs), is sized at 6,14 gibibytes in total.

Due to the mechanism of only loading necessary the tiles and its neighbours, the cache does not become flooded with unnecessary data. The average cache that was required for each substance was tested by using the web-application on a standard Mozilla Firefox web-browser. During this test the 6th region was taken as the center point resulting in tiles 5/6/7 being loaded for the current week and the weeks before/after. Table 2 shows the memory cached whilst showing different weeks and substances.

## 5.2 General visualisations

As was explained in section 4.4.2 some distortion was expected among the poles. This distortion is visible in figure 9. The closer one gets to the equator the closer points get to each other. No COVID-19 data is available around the poles so in the scope of correlation between COVID and air pollution this does not cause any large problems.

| Folder name | Original size | Application size |
|---|---|---|
| L2__NO2___/2019 | 711 GiB | 0.714 GB |
| L2__NO2___/2020 | 940 GiB | 0.719 GB |
| L2__NO2___/2019 | 711 GiB | 0.714 GB (S) |
| L2__NO2___/2020 | 940 GiB | 0.719 GB (S) |
| L2__SO2___/2019 | 1567 GiB | 0.716 GB |
| L2__SO2___/2020 | 2008 GiB | 0.721 GB |
| L2__CH4___/2019 | 109 GiB | 0.242 GB |
| L2__CH4___/2020 | 121 GiB | 0.198 GB |
| L2__CO____/2019 | 267 GiB | 0.715 GB |
| L2__CO____/2020 | 340 GiB | 0.726 GB |
| **Total** | **7714GiB** | **6.2GiB** |

**Table 1: Data reduction, (S) indicates stratosphere**

| Substance | Week 2 | Week 12 | Week 23 |
|---|---|---|---|
| NO2 | 81.90 MB | 98.30 MB | 67.30 MB |
| SO2 | 78.65 MB | 98.50 MB | 70.70 MB |
| CH4 | 33.76 MB | 27.70 MB | 21.40 MB |
| CO | 82.11 MB | 90.80 MB | 70.40 MB |

**Table 2: Cached memory**

## 5.3 Substance visualisations

Each of the substances has, as discussed in section 4, it's own colormap and value range. The results therefore vary greatly between the substances. For each substance an image is taken of Europe and of China. All images of Europe are taken in week 12 because there is a clear difference visible between 2019 and 2020 and Covid-19 restrictions were quite high. All images from China were taken in week 6 since at that point China's lockdown restrictions were high. These images can be found in the appendix. In figure 10 and figure 17 some visualisations in respectively China and Europe can be seen for the tropospheric vertical column of $NO_2$. The full spectrum of colors is visible on each of the pictures and some differences are visible for the images between 2019 and 2020. Figure 12 and figure 13 show the stratospheric column of $NO_2$. Only a few of the colors are visible of the spectrum and differences between 2019 and 2020 are harder to notice. There are also some results for the $SO_2$ dataset. These can be seen in figure 14 and 15. Just like $NO_2$ all colors are visible, but high and low values are directly next to each other on a lot of places. The visualisations that were made of CH4 look quite different. There are a lot of gaps visible on the map as can be seen in figure 16 and figure 17. Most values also fall into the orange spectrum with only some slight deviations visible to red and yellow. Lastly, there is CO. When looking at China in figure 18 there are some distinct orange features visible, but no red ones. Most values fall into the blue range. This is even more accentuated when looking at Europe in figure 19. There is only blue to be found on this picture.

# 6. CONCLUSIONS

As was stated in 3 the goal of this project was to contribute to the TROPOMI objectives by answering the question '*How can the reduction of global greenhouse gas emissions during the COVID-19 pandemic be visualized concisely and interactively?*'. The visualisation on the webpage shows the 'how' of this research question. To do this concisely has been achieved through the thorough analysis, extraction and compression of the data and translating that to coordinates that received a certain value, which in turn could be visualised in a manner that is easy to understand. The interactive part of the research question is evident through the possible interactions the user can have. The idea of this is that allowing the user to interact and search for places to see the evolution of greenhouse gases, an engaging experience might make some people curious.
The sub questions can also be found in the visualisation. Sub-question 1 is related to the choice of showing data per week instead of, for example, by day or hour. Then, for the visualisation of the data itself the obvious choice is to show the emission per area in a colored scale in which a upper and lower limit indicate if the emission was high or low related to the range of measured values. To answer the next sub-question it was necessary to look somewhere else for an answer. As has been stated before (see 4.7) an external source was used to integrate lockdown strictness into the visualisation. Lastly, showing the data to the end-user in a user friendly way was done by chunking the data into smaller files and only keeping data in a users cache that is required for that current visualisation. By doing this the client browser does not have to render all the files at the same time, thus enhancing the load time and usability of the visualisation.

# 7. DISCUSSION

During the course of this project some problems and errors occurred as well as some actions were taken that require elaboration. For the future, and if this project will be continued, taking this discussion into consideration is essential to ensure correct and usable results will be achieved and improvements of the data and analysis can be made.
Firstly, it must be stated that the people working on this project are by no means experts in chemistry, geology, meteorology or any other field related to the content of the data. Therefore, at times it was difficult to make sense of the data and use correct parameters in the source code. Most significantly this came forward during the creation of the color maps. As has been explained in section 4.6, for each color map a range was chosen based either on research or the guides accompanying the TROPOMI [23], but the found numbers were not always equal. Furthermore, to create the weekly tiles the data was averaged, which resulted in a loss of data. If these ranges, color scales and averages to visualize the data make sense and were implemented correctly should be discussed with an expert, preferably from the Copernicus Institute of Sustainable Development. It was also noted that the color maps were in some cases less suitable for a visualisation. Most color maps were first tested on a few samples, this was especially the case for CO. As can be seen in the CO example, this might not have been the best color map to pick globally. Most regions can be quite monotone in color. The averaging that was chosen also caused some distortion as can be seen in 9. This is of course not ideal, if the visualisations would be made again it might be worthwhile looking at a different approximation algorithm.
Once we started running the algorithm to extract the data from the NetCDF files we ran into certain errors indicating that coordinates were either out of bounds or corrupt. This has been discussed in section 4.4.2. To make the results more reliable, more research should be done into finding the source of these errors, perhaps again with an expert from the Copernicus Institute. Reducing the amount of corrupt files would decrease the missing information. After running the algorithm it was found in some cases that either no information was written or information was written to the incorrect files. The reason for this most probably was human error, since before running the algorithm changing the global variables as well as adjusting the files to be written to had to be done manually. These files were later corrected to ensure no incorrect data was used in the visualisation, however this caused more budget to be used then necessary.
Lastly, it is worth mentioning that extending this research to identify local and global trends in the emission would be very interesting. During this project this did not happen, even though the year 2018 could have been added. Due time and resource constraints it was decided not to add another year to each substance, to ensure that as many substances as possible could be added to the visualisation.

# References


[1] R. Baldwin and B. Weder di Mauro. Economics in the time of covid-19, 2020.

[2] D. Harrington and M. Hadjiconstantinou. Commuting behaviours and covid-19, 2020.

[3] H. Akimoto. Global air quality and pollution. *Science*, (302(5651)):1716–1719, 2003.

[4] V. Ramanathan. Air pollution, greenhouse gases and climate change: global and regional. *Atmospheric Environment*, (43(1)):37–50, 2009.

[5] L. Menut, B. Bessagnet, G. Siour, S. Mailler, R. Pennel, and A. Cholakian. Impact of lockdown measures to combat covid-19 on air quality over western europe. *Science of the Total Environment*, 741:140426, 2020.

[6] Z. S. Venter, K. Aunan, S. Chowdhury, and J. Lelieveld. Covid-19 lockdowns cause global air pollution declines. *Proceedings of the National Academy of Sciences*, 117(32):18984–18990, 2020.

[7] J. Veefkind, I. Aben, K. McMullan, H. Förster, J. De Vries, G. Otter, J. Claas, H. Eskes, J. De Haan, Q. Kleipool, et al. Tropomi on the esa sentinel-5 precursor: a gmes mission for global observations of the atmospheric composition for climate, air quality and ozone layer applications. *Remote Sensing of Environment*, 120:70–83, 2012.

[8] M. Bauwens, S. Compernolle, T. Stavrakou, J.-F. Müller, J. Van Gent, H. Eskes, P. F. Levelt, R. van der A, J. Veefkind, J. Vlietinck, et al. Impact of coronavirus outbreak on no2 pollution assessed using tropomi and omi observations. *Geophysical Research Letters*, 47(11):e2020GL087978, 2020.

[9] C. Wang, T. Wang, P. Wang, and V. Rakitin. Comparison and validation of tropomi and omi no2 observations over china. *Atmosphere*, 11(6):636, 2020.

[10] S. Sannigrahi, A. Molter, P. Kumar, Q. Zhang, B. Basu, A. S. Basu, and F. Pilla. Examining the status of improved air quality due to covid-19 lockdown and an associated reduction in anthropogenic emissions. *medRxiv*, 2020.

[11] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers. A description of the advanced research WRF version 2. Technical report, National Center For Atmospheric Research Boulder Co Mesoscale and Microscale . . ., 2005.

[12] S. Mailler, L. Menut, D. Khvorostyanov, M. Valari, F. Couvidat, G. Siour, S. Turquety, R. Briant, P. Tuccella, B. Bessagnet, et al. Chimere-2017: from urban to hemispheric chemistry-transport modeling, 2017.

[13] M. Ghahremanloo, Y. Lops, Y. Choi, and S. Mousavinezhad. Impact of the covid-19 outbreak on air pollution levels in east asia. *Science of The Total Environment*:142226, 2020.

[14] Global nitrogen dioxide monitoring home page. https://so2.gsfc.nasa.gov/no2/no2_index.html. Accessed:2020-9-27.

[15] J. Wang, M. Chen, G. Lü, S. Yue, K. Chen, and Y. Wen. A study on data processing services for the operation of geo-analysis models in the open web environment. *Earth and Space Science*, 5(12):844–862, 2018.

[16] S. Steiniger and A. J. Hunter. Data structure: spatial data on the web. *International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology*:1–12, 2016.

[17] L. van den Brink, P. Barnaghi, J. Tandy, G. Atemezing, R. Atkinson, B. Cochrane, Y. Fathy, R. Garcıa Castro, A. Haller, A. Harth, et al. Best practices for publishing, retrieving, and using spatial data on the web. *Semantic Web*, 10(1):95–114, 2019.

[18] G. Romanillos, B. Moya-Gómez, M. Zaltz-Austwick, and P. J. Lamıquiz-Daudén. The pulse of the cycling city: visualising madrid bike share system gps routes and cycling flow. *Journal of Maps*, 14(1):34–43, 2018.

[19] R. Page. Visualising geophylogenies in web maps using geojson. *PLoS currents*, 7, 2015.

[20] P. Butler. Mapping temporal datasets with d3. *Cartographic Perspectives*, (81):44–48, 2015.

[21] Cloud optimized geotiff. https://www.cogeo.org/. Accessed:2020-9-27.

[22] R. Rew and G. Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

[23] S5p mission performance centre nitrogen dioxide readme. https://sentinels.copernicus.eu/documents/247904/3541451/Sentinel-5P-Nitrogen-Dioxide-Level-2-Product-Readme-File. Accessed:2020-9-27.

[24] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub, et al. The geojson format. *Internet Engineering Task Force (IETF)*, 2016.

[25] S. Talukdar, S. Mahato, S. Pal, S. Debanshi, P. Das, A. Rahman, et al. Modelling the global air quality conditions in perspective of covid-19 stimulated lockdown periods using remote sensing data, 2020.

[26] M. Saunois, A. R. Stavert, B. Poulter, P. Bousquet, J. G. Canadell, R. B. Jackson, P. A. Raymond, E. J. Dlugokencky, S. Houweling, P. K. Patra, et al. The global methane budget 2000–2017. *Earth System Science Data*, 12(3):1561–1623, 2020.

[27] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[28] B. N. Duncan, L. N. Lamsal, A. M. Thompson, Y. Yoshida, Z. Lu, D. G. Streets, M. M. Hurwitz, and K. E. Pickering. A space-based, high-resolution view of notable changes in urban nox pollution around the world (2005–2014). *Journal of Geophysical Research: Atmospheres*, 121(2):976–996, 2016.

[29] Tropomi data-products. http://www.tropomi.eu/data-products. Accessed:2020-10-21.

[30] Earth observatory nasa so2 pollution controls bring results. https://earthobservatory.nasa.gov/images/76571/so2-pollution-controls-bring-results. Accessed:2020-10-21.

[31] Earth observatory nasa sulfur dioxide down over china up over india. https://earthobservatory.nasa.gov/images/87154/sulfur-dioxide-down-over-china-up-over-india. Accessed:2020-10-21.

[32] G. M. Murch. Physiological principles for the effective use of color. *IEEE Computer Graphics and Applications*, 4(11):48–55, 1984.

[33] C. Frankenberg, J. F. Meirink, M. van Weele, U. Platt, and T. Wagner. Assessing methane emissions from global space-borne observations. *Science*, 308(5724):1010–1014, 2005.

[34] A. Turner, D. J. Jacob, K. J. Wecht, J. D. Maasakkers, E. Lundgren, A. E. Andrews, S. C. Biraud, H. Boesch, K. W. Bowman, N. M. Deutscher, et al. Estimating global and north american methane emissions with high spatial resolution using gosat satellite data, 2015.

# 8. APPENDIX

## 8.1 Webpage

The data for the webpage that was created can be found on databricks in the following directory: dbfs:/mnt/group08/website

## 8.2 Result visualisations

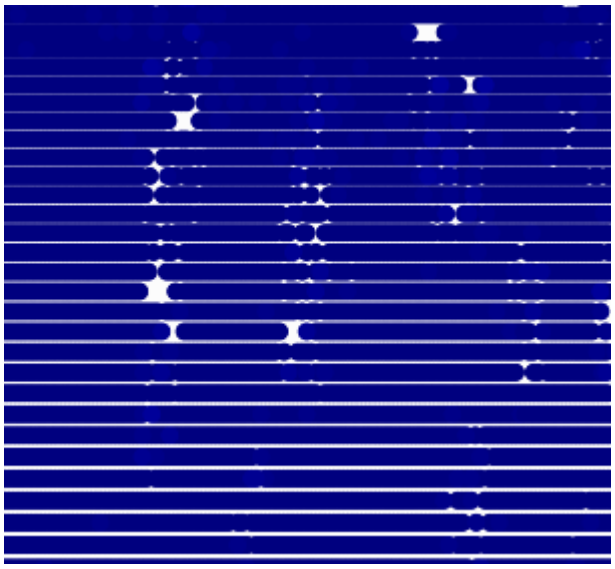

**Figure 9: Averaging causes heavy distortion among the poles**

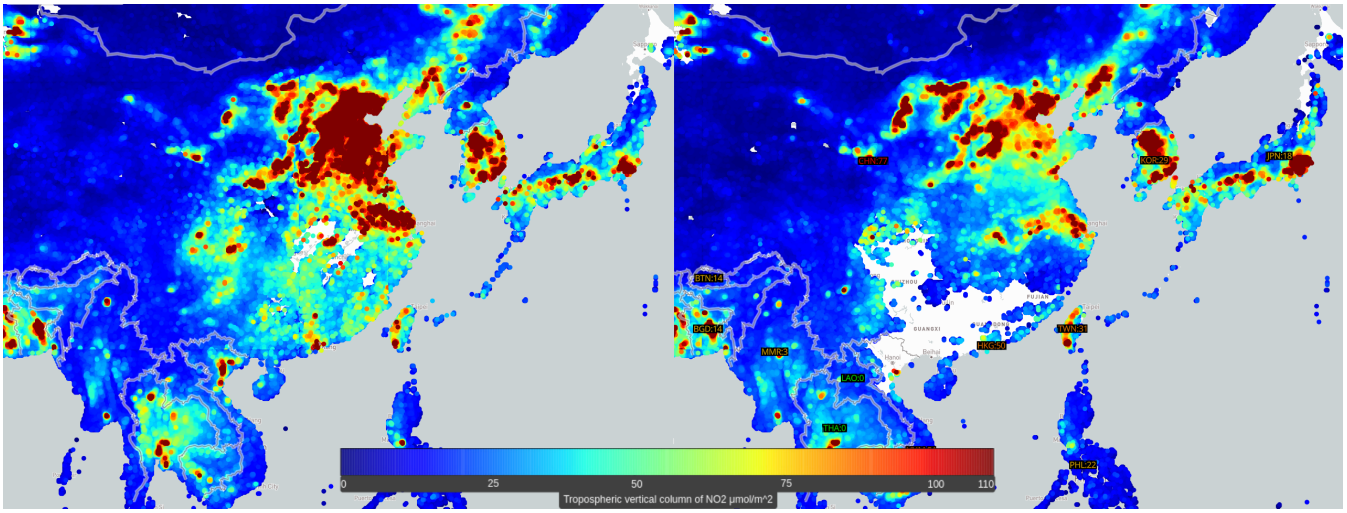


**Figure 10: NO2 tropospheric column in 2019 and 2020 week 6 in China**

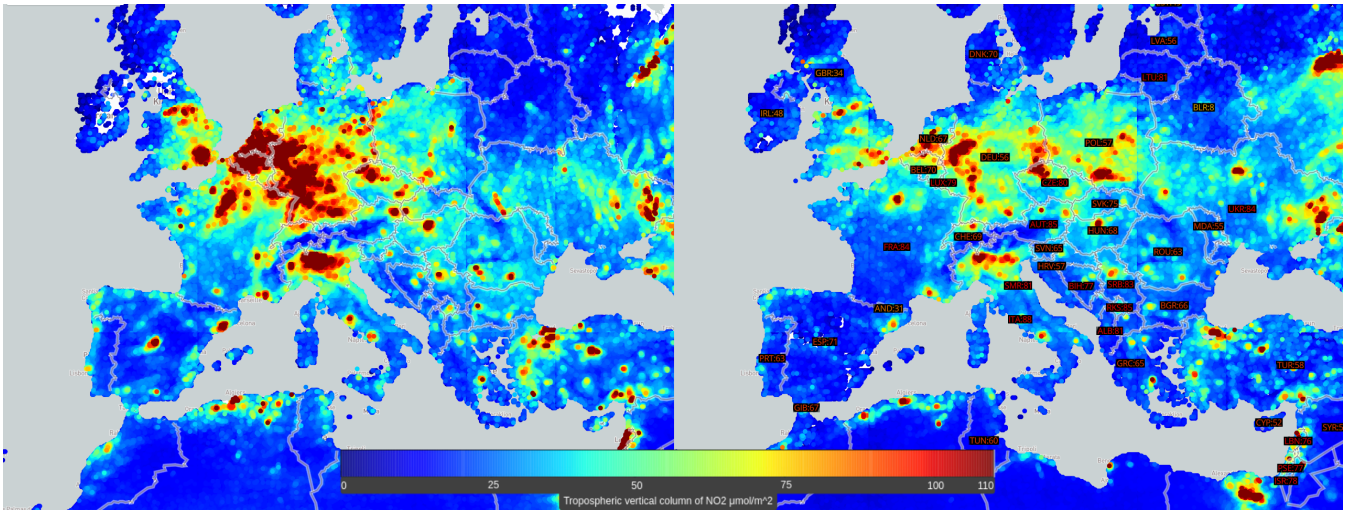


**Figure 11: NO2 tropospheric column in 2019 and 2020 week 12 in Europe**

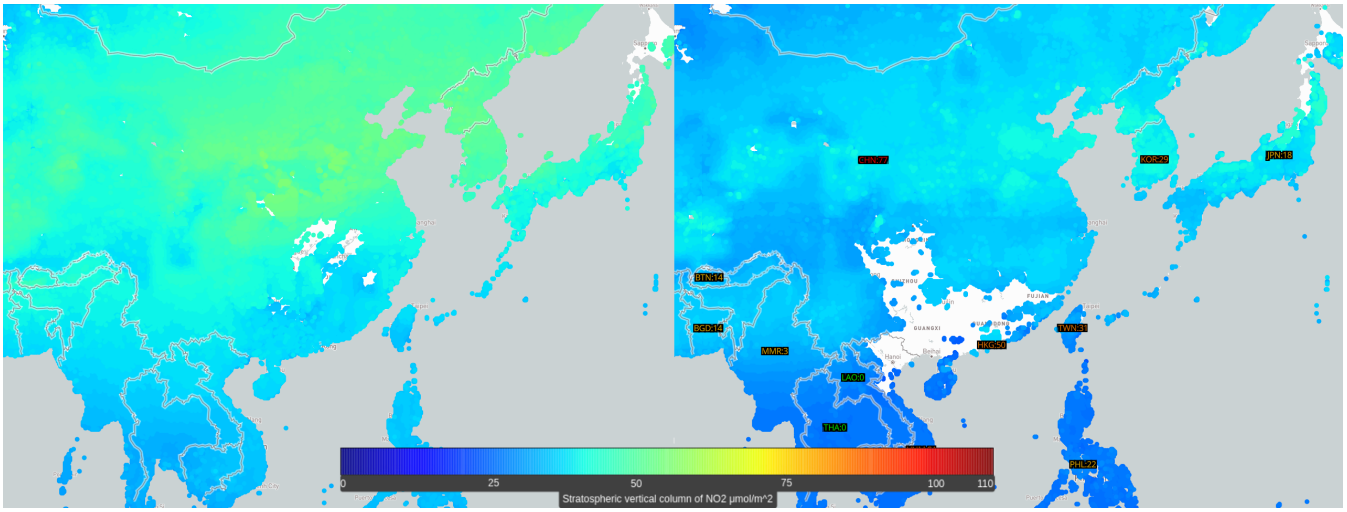


**Figure 12: NO2 stratospheric column in 2019 and 2020 week 6 in China**

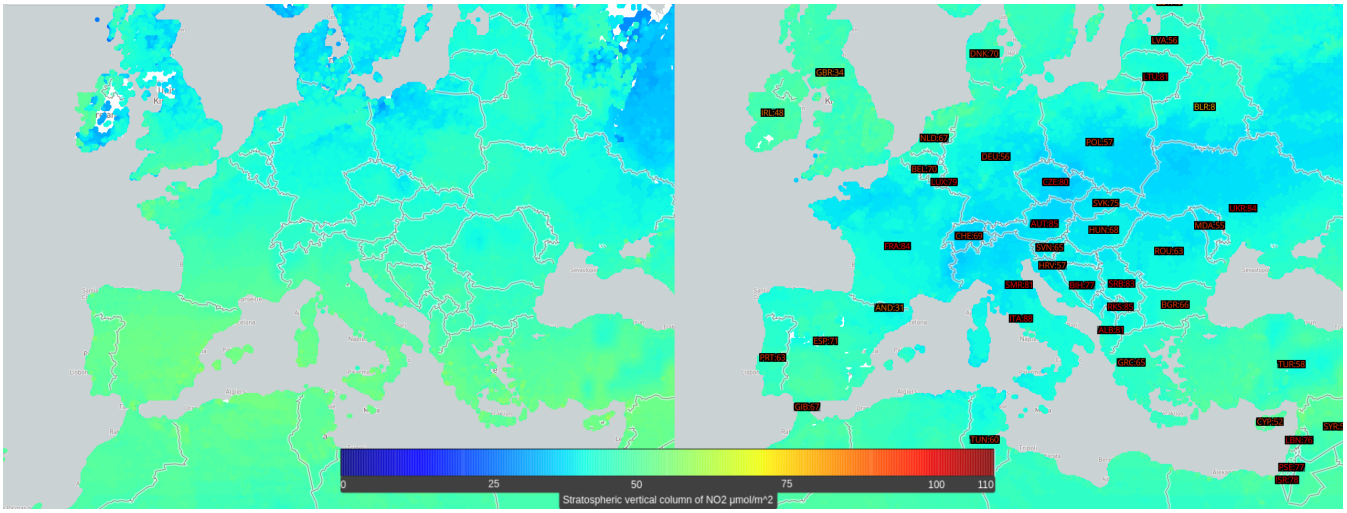


**Figure 13: NO2 stratospheric column in 2019 and 2020 week 12 in Europe**

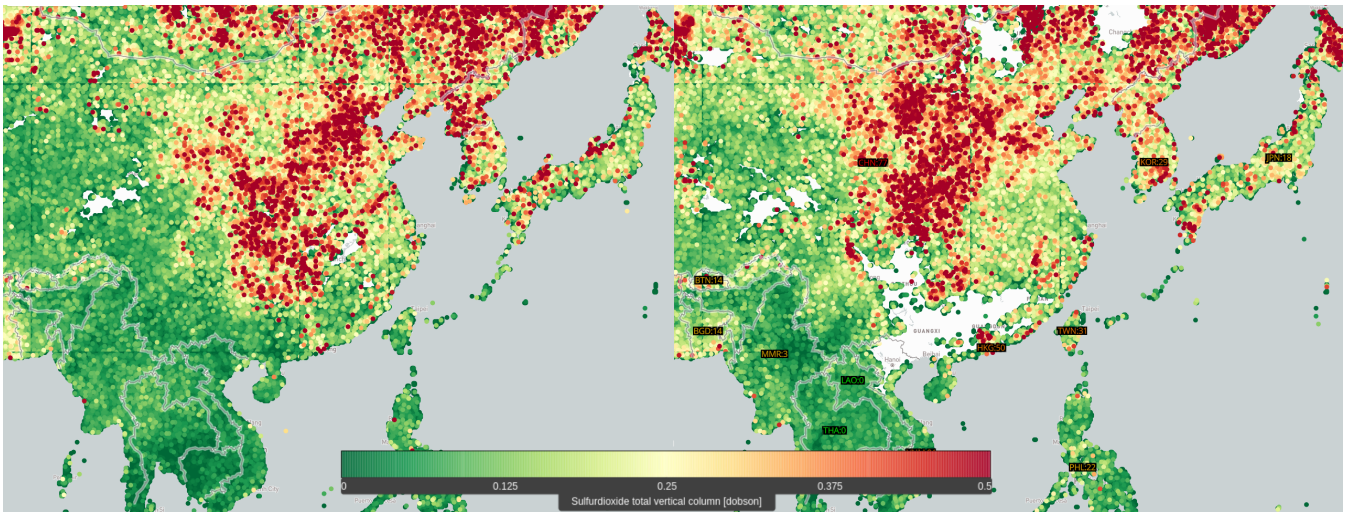


**Figure 14: SO2 total column in 2019 and 2020 week 6 in China**

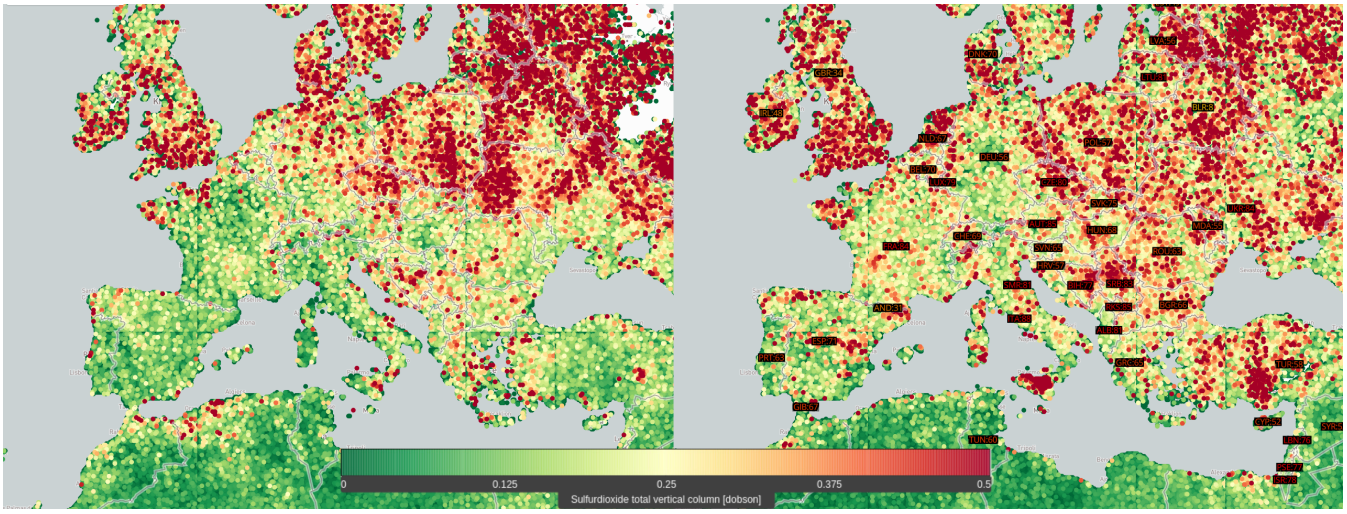


**Figure 15: SO2 total column in 2019 and 2020 week 12 in Europe**

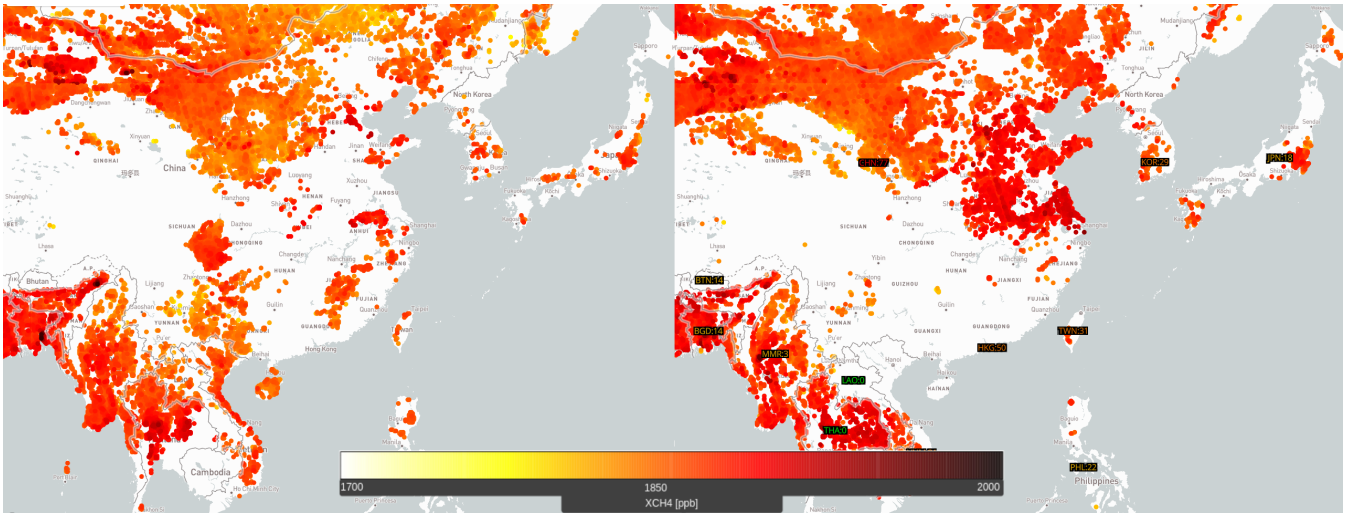


**Figure 16: xCH4 mixing ratio in 2019 and 2020 week 6 in China**

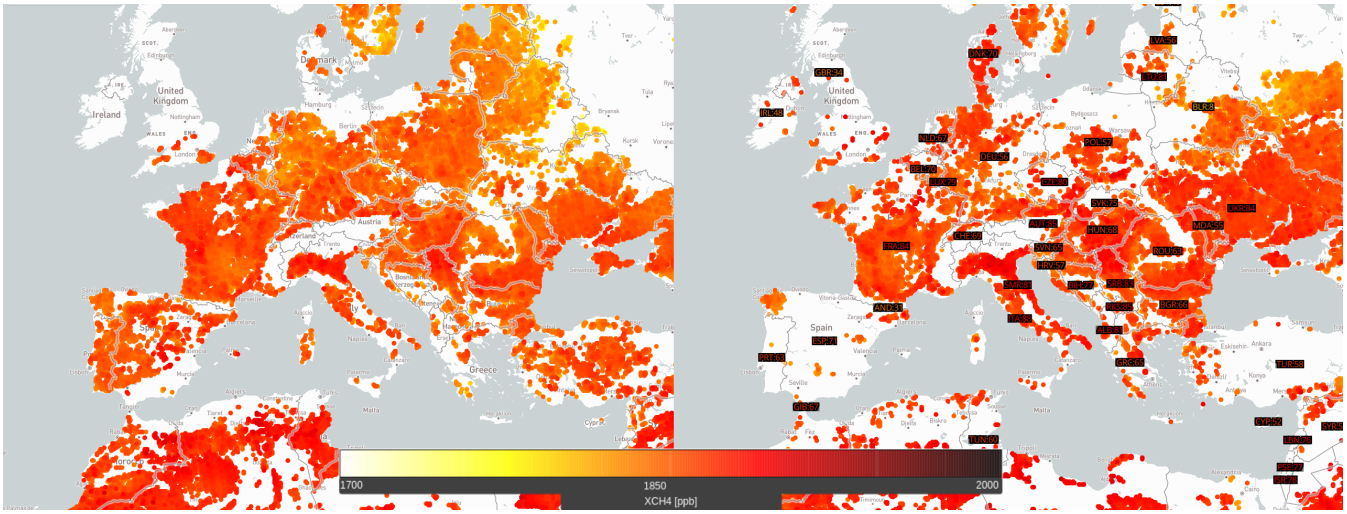


**Figure 17: xCH4 mixing ratio in 2019 and 2020 week 12 in Europe**

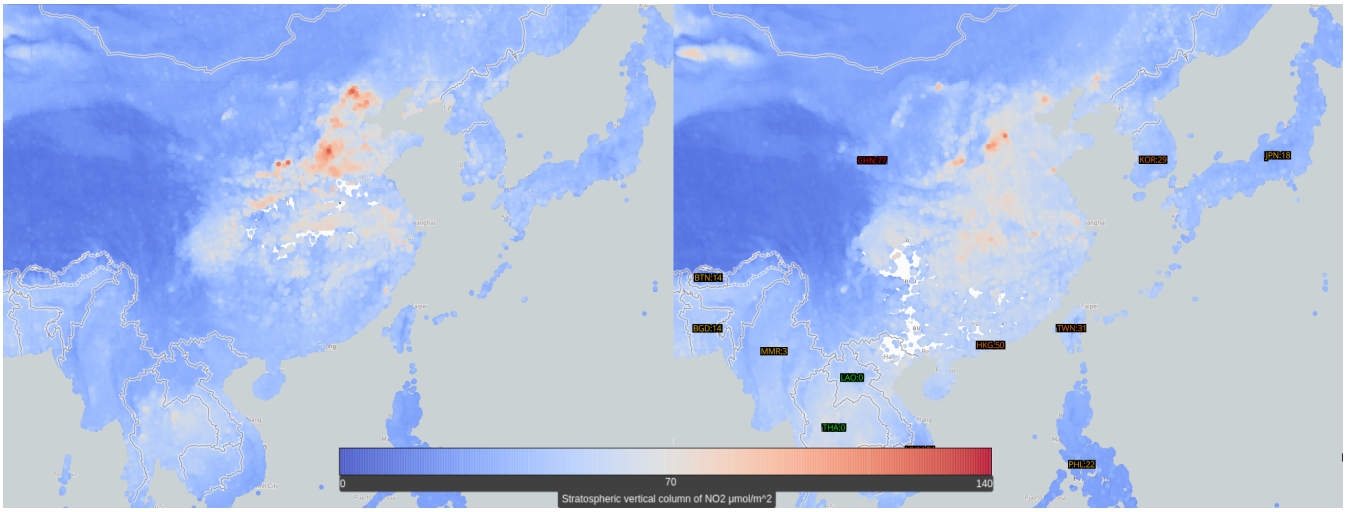


**Figure 18: CO total column in 2019 and 2020 week 6 in China**

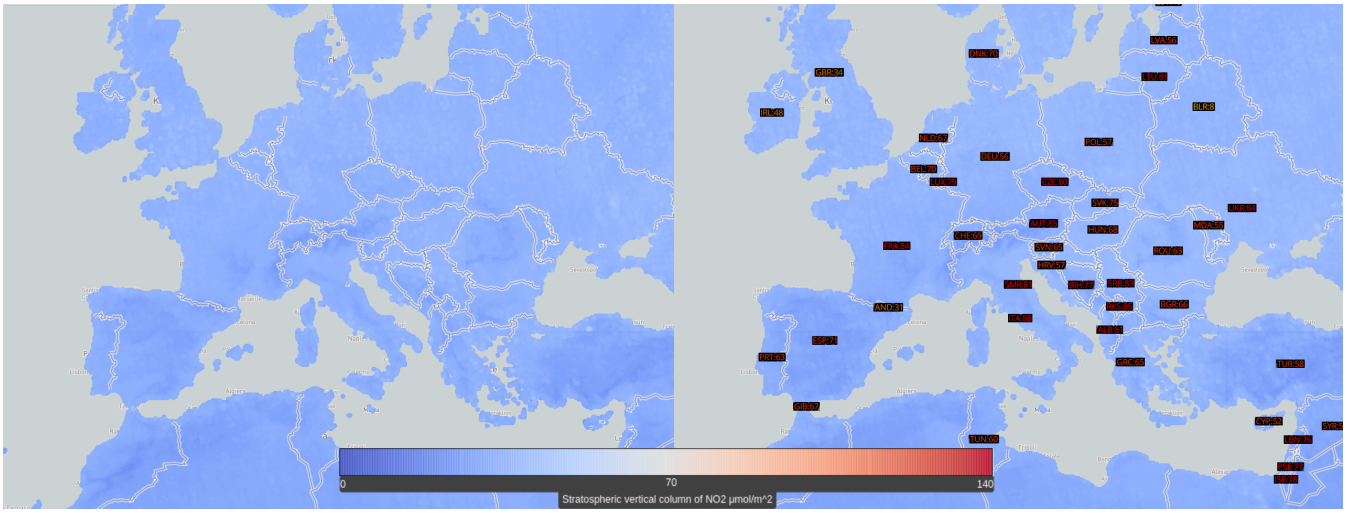


**Figure 19: CO total column in 2019 and 2020 week 12 in Europe**

| **Task** | **Who** |
|---|---|
| Initial data investigation | All |
| Project plan | All |
| Test visualisations with Cartopy | Krijn |
| Colormaps and minimum and max values | Krijn. Tim helped with determining the range. Further on all colormaps were debated and improved upon team feedback. |
| Visualisation and layout of colorbars based on colormaps | Lucas |
| Local pipeline | Krijn |
| TaskCreator.py | Krijn |
| Parallelizing the code on Databricks (spark) | Krijn and Lucas |
| Pipeline: Averaging data | Krijn and Lucas |
| Pipeline: Tiling, Assigning colors | Krijn |
| Pipeline: handling corrupt data | Tim and Lucas |
| Determining what was necessary for the GeoJSON format | Tim |
| Determining how to keep the GeoJSON format compact | All |
| Writing GeoJSON | Krijn |
| Optimizing the pipeline | Krijn and Lucas |
| Filtering data and defining metrics | All |
| COVID-19 data investigation | All |
| COVID-19 pipeline | Lucas |
| COVID-19 GeoJSON and writing | Lucas |
| COVID-19 foreground colors | Krijn |
| Running queries on DataBricks and debugging | All, but mainly Tim and Lucas |
| Site hosting and managing | Lucas |
| Investigation in visualisation tools | Tim |
| Setting up initial folium and mapbox environments and creating preliminary visualisation | Tim |
| Javascript:<br>Allowing changes in weeks/substances<br>"Chunkloading" | Lucas |
| Website layout (other than the basic Mapbox Map) - CSS /Javascript | Lucas |
| Site displaying 2 maps at the same time | Krijn |
| Synchronising the 2 maps and handling input | Tim |
| Report sections: The pipeline, substances, the JSON formats, Results for substances, everything about colormaps (project plan, conclusion, discussion) | Krijn |
| Report sections: Abstract/Introduction/Related works/Research Question (partly), Data sizes for Input Data, COVID-19 Data, Visualisation (partly), Results data reduction, Discussion (partly). | Lucas |
| Checking, adding sources and finding these sources | All |
| Report sections: Abstract/Introduction/Related works/Research Question (mostly), Raw input data, initial data investigation, Visualisation (mostly), Conclusion, Discussion (mostly) | Tim |
| Checking, adding sources and finding these sources | All |
| Final formatting of the document and deliverables | Tim |
| Presentation: Intro, RQ, Scope, First results | Tim |
| Presentation: Pipeline, GeoJSON | Krijn |
| Presentation: COVID-19 Data, Demonstration, Conclusion | Lucas |