# Information Visualization assignment

Marco Zanetti

# 1 Introduction

In the following report I explain my work with respect to the visualization project for the Information Visualization course. I start by explaining the dataset and the question answered by my visualization. I will then talk about the tools I used and why I choose them. I will talk about some challenges I encountered during the project and how I solved them, and then I'll present the final visualization and explain how to run it. A brief conclusion will then wrap up the work.

# 2 The problem to solve

The dataset at our disposal describe the work travels by the members of a research lab. This dataset is composed by four *.tsv* files, respectively:

1 **missions.tsv**: trips details;

2 **users.tsv**: members of the lab (pseudonymized);

3 **places.tsv** : travel destinations;

4 **countries.tsv**: country codes.

The four datasets are correlated trough the uses of primary and foreign keys.

## 2.1 The question

The question I choose to solve is: **How are emissions distributed across the regions?**. In particular, we would like to know how the Co2 emissions are distributed across the regions over time.

## 2.2 Why is this interesting?

Regions in the dataset describe the different teams in the labs. Trough the proposed visualization, we can study which regions are most active (they are probably going to travel more), how the trips are distributed over the years (are they constant? were there any abrupt changes or different trends?), which are the preferred transportations for each team.

## 2.3 The proposed solution

The chart will have the cumulative emissions over time on the $Y$ axis and the date on the $X$ axis. Colours are going to be used to describe the different regions. A legend on the right side of the chart help mapping the colours with the regions, while also allowing to filter the active regions on the visualization. Detailed information are showed when moving the cursor over a particular line. A notable introduction from the previously proposed visualization is a filter that allows to discriminate cumulative emissions for different types of transportations added trough the use of a check-list.

## 2.4 The tools

As a programming language, I relied on *Python* for its vast support by the community (many libraries available), its simplicity of use and most importantly my previous experience with it. I also used the following external libraries, all present on *pypi*:

- **Plotly**: An open-source interactive data visualization;
- **Plotly express**: An high level wrapper for Plotly;
- **Dash**: A python framework for building reactive web-apps;
- **Jupyter-dash**: Dash support for Jupyter Notebook interface;
- **Jupyter Lab**: A web based environment for interactive computing;

- **Pandas**: Popular library for data structures;
- **Pathlib**: Object-oriented file system path.

*Jupyter Lab* was used for developing purposes due to its effectiveness when debugging. While a Jupyter file is also provided, the whole project is also translated in single python file to run. *Plotly* is the key visualization component. I chose to use plotly for my interactive visualization framework for its simplicity of use and the fact that is a popular open source option. Powerful visualization can be conceived trough little lines of code, so much that I found possible to add even more functionality that I had originally planned. The Plotly graph is then showed in the browser trough *Dash*, a python framework for building web applications. A Dash application is composed by the layout, which describes how the application will look like, and a second section which describes the interactivity of the application.

# 3 Challenges

Here I'll present some notable challenges that I faced and how I proceeded to solve them. With regards to the code, I will focus on some particular 'interactions' in the frameworks that I had to discover and tweak for my visualization to work. I will also mention some questions that I posed myself regarding some aspects of the visualization.

1 **How can we approximate the 'other' transportation method in the graph**: As I mentioned before, I decided to use a check-list to discriminate total emission for single transportation methods or a subset of them. There are four methods: *plane, train, car* and *public*. A fifth entry on the dataset, *other*, describe unregistered methods. To visualize this, there are two possible options: The first one consists in display *other* in the check-list, and therefore considering the uncertain method *other* as a new category. The second one (the one I opted for) consists in using the approximated Co2 consumes computed using the algorithm and values from DEFRA 2016 to infer back the approximated method. Since we are already relying on this approximation for obtaining the Co2 values, i found fitting to use this also for the method.

2 **A visualization issue occurred when displaying time-series values** As can be seen in figure [1], an interesting issue occurred when displaying the results, where values where 'jumping forward and backward in time'. I then discovered that it was required for the Plotly *px.line()* function to work properly with time-series that the dates in the dataset should be ordered. I solved this issue by ordering the dataset for date instead that for user id as it was originally.

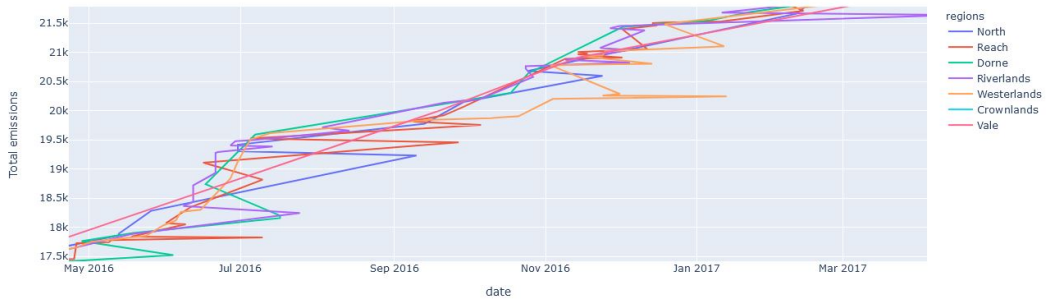


Figure 1: Time-series visualization error

3 **The lines appears as a single growing line**: This was yet another interaction problem using Plotly. To plot the cumulative emissions, I had to use the *.cumsum()* python function to get a growing value going forward in time (without this cumulative sum, the output was as in figure [2]). However, since the regions discrimination was done externally, it was summing the values of each region together, thus creating the visualization seen in figure [3]. I solved this issue by creating a new column for cumulative emissions grouping by regions beforehand.

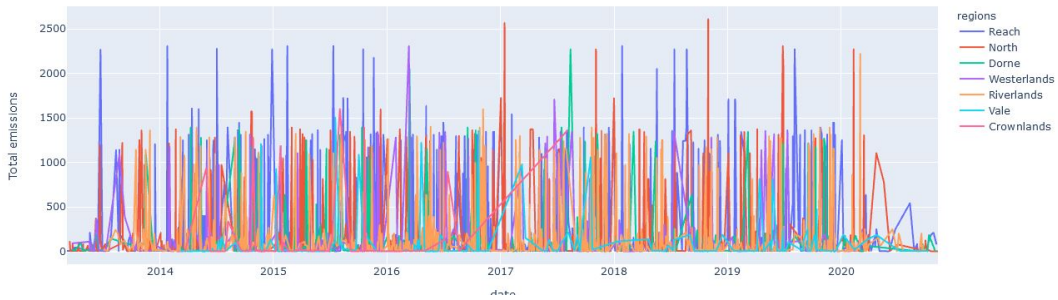


Figure 2: Visualization without using the cumulative sum

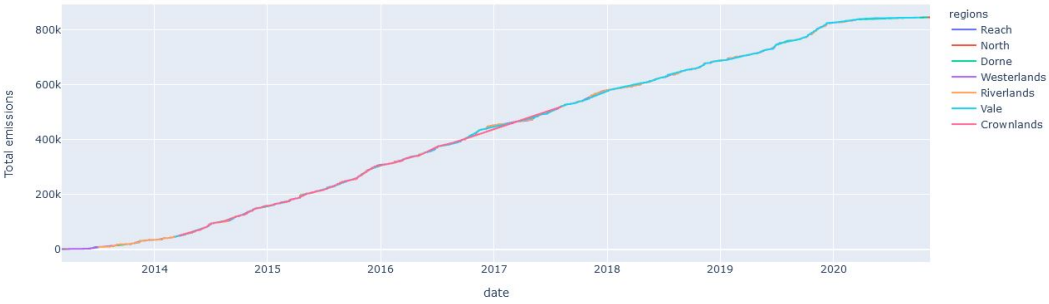


Figure 3: Visualization without using the groupby beforehand error

4 **The color was changing constantly when using the check-list**: Using the automatically assigned Plotly colors in the *px.line()* function cause the colors to be mapped not for region but for the order they appears in the dataframe. This may, for example, cause a region that may be a certain color (when considering all the transportation methods) to change its color when only considering the *car* method. This may obviously disorient and confuse the user. To solve this problem, I assigned a static mapping of each region with the colors.

5 **The choice of colours**: Colours are an important component in this visualization, since they help the user map each region in its respective line in the graph. I therefore decided to account for colour-blind users by using the *Okabe-Ito* palette proposed by Masataka Okabe and Kei Ito. I chose this palette due to its use of appealing color for the wider audience while also accounting for colour-blind people. The seven color used are *orange, sky blue, bluish green, yellow, blue, vermillion* and *reddish purple.*

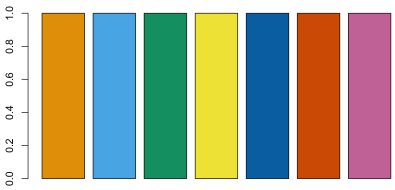


Figure 4: The Okabe-Ito palette

# 4 The project structure

The folder structure requires all the datasets in a folder called *Data*, while the code is found in the main folder. The structure of the code is pretty straightforward. The first thing to do is merging all the datasets so that we have all the information in a single place. Since the function to do this tasks take several minute, the dataset gets generated (and saved) only when it's not already present on the disk, otherwise it's just loaded. The main code is span trough the layout section, where I define the check-list and the graph. The *update_line_chart* sets the fixed colours, gets the cumulative emissions, filter for the transportation selected trough the check-list and then display the visualization using the *px.line()* function. Here we explicit the input data, we name the labels, we associate the colors and group lines by regions. The final visualization is showed in figure [5]. It can either be used in Jupyter or trough the python script. The script can be run using the command *python viz.py* in the command line. The visualization will then be loaded in the browser at the *http://127.0.0.1:8050/* local address. Using

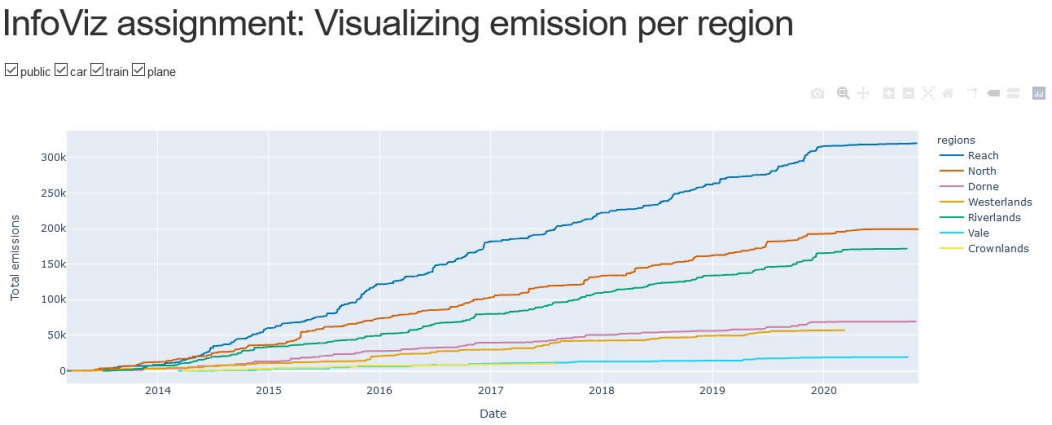


Figure 5: Definitive visualization look

the check-list and the legend, we can obtain powerful and clear insight. For example, figure [6] shows how we can easily compare car emissions for two selected regions. Other than the filtering provided by

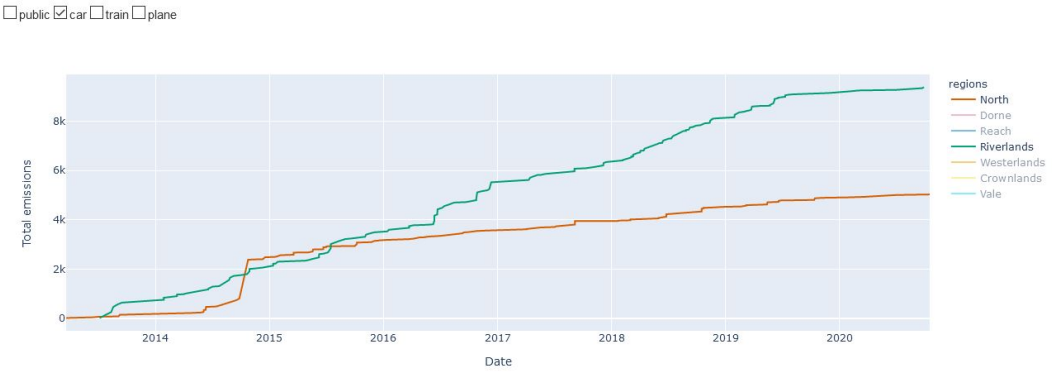


Figure 6: Visualization insight example

the check-list and the legend, Plotly automatically implement a zoom option, spikes lines and mouse hovering, allowing for even greater interaction and clarity. An example of this is showed in figure [7].

# 5 Comments

I was able to extend from what I conceived before in the 'design' phase since the Plotly library offered me a lot of options that I was not expecting. In particular, the zoom function and the methods check-

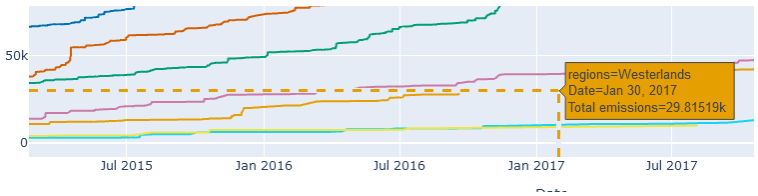


Figure 7: Plotly additional feature showcase

list were not in my initial design. In my opinion, this visualization effectively answers the question I selected (**How are emissions distributed across the regions?**). The colors and the legend makes the comparison intuitive and easy to see both if we are interested in single regions or comparisons. The check-list add another useful filter that allows for greater insights and it's simple to use for the user. The visualization also offers extra interactions such as zoom and mouse hovering if needed.

# 6 Possible improvements and conclusion

Trough the visualization presented in this report, I was able to answer the question proposed previously while also further improve upon it with additional functionalities. Some possible improvements sees the introduction of some clarity for the user regarding the *other* approximation cited previously and a better looking check-list. Overall I feel like the visualization fully conveyed the answer to the original question posed at the start of this report, while also following visualization rules such as the visual information-seeking mantra (overview, zoom and filter and then details on demand) and colors rules (no more than 7, adapt for color-blind).