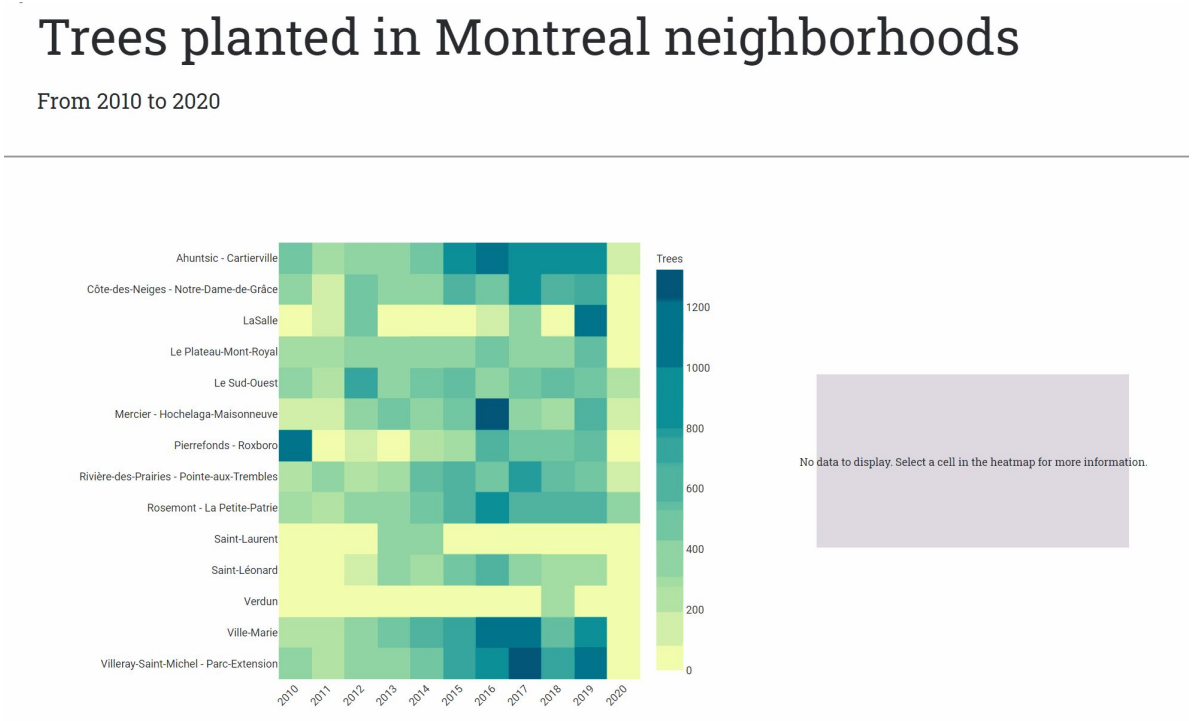
Guide TP3

INF8808E | Summer 2022

Version Python

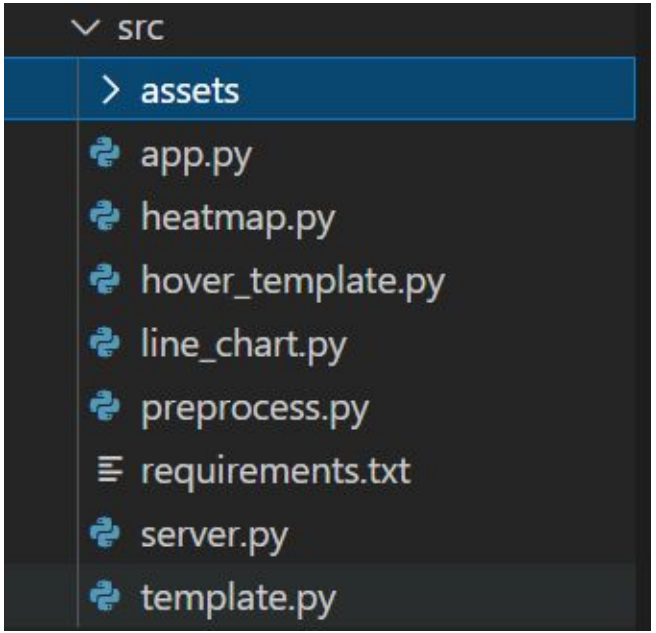
Objective

- The goal of this lab is to create an interactive heatmap and line chart dashboard using open data in CSV format.



General Info

File Structure

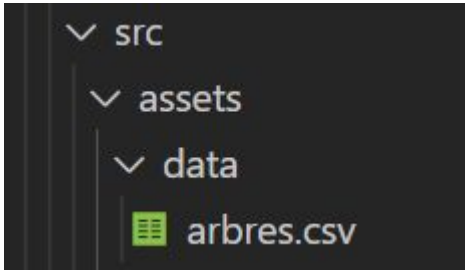


- Create the *venv* and install *requirements.txt*
- You should not modify the files ***app.py*** et ***server.py***
- You have to fill the different sections **TODO** on the files

Dataset

Plantations of trees in Montréal

- Data represent tree plantations over time in Montreal neighborhoods,
- It contains data representing the date and location that public trees were planted in Montreal,
- You will find the dataset on: assets -> data -> arbres.csv



- `Arrond` : The ID of the neighborhood where the tree was planted.
- `Arrond_Nom` : The name of the neighborhood where the tree was planted.
- `Date_Plantation` : The date the tree was planted.
- `Longitude` : The longitude of the tree.
- `Latitude` : The latitude of the tree.

Data processing

Reorganize certain parts of it so they can be properly used by the Plotly library

- Convert the dates in the data to a format understandable by Plotly - function ***'convert_dates'***
- Filter the data by year (2010-2020) - function ***'filter_years'***
- Summarize the data to get the total count of trees planted per year per neighborhood - function ***'summary_yearly_counts'***
- Restructure the data into a format that Plotly can easily read to generate the heatmap - function ***'restructure_df'***
- Write a function to easily get data to display on the line chart, corresponding to the daily amount of planted trees in a given neighborhood and year - function ***'get_daily_info'***

Data processing

Figure 1 illustrates data in a format that can be displayed in the heatmap, while Figure 2 samples some data that can be displayed in the line chart

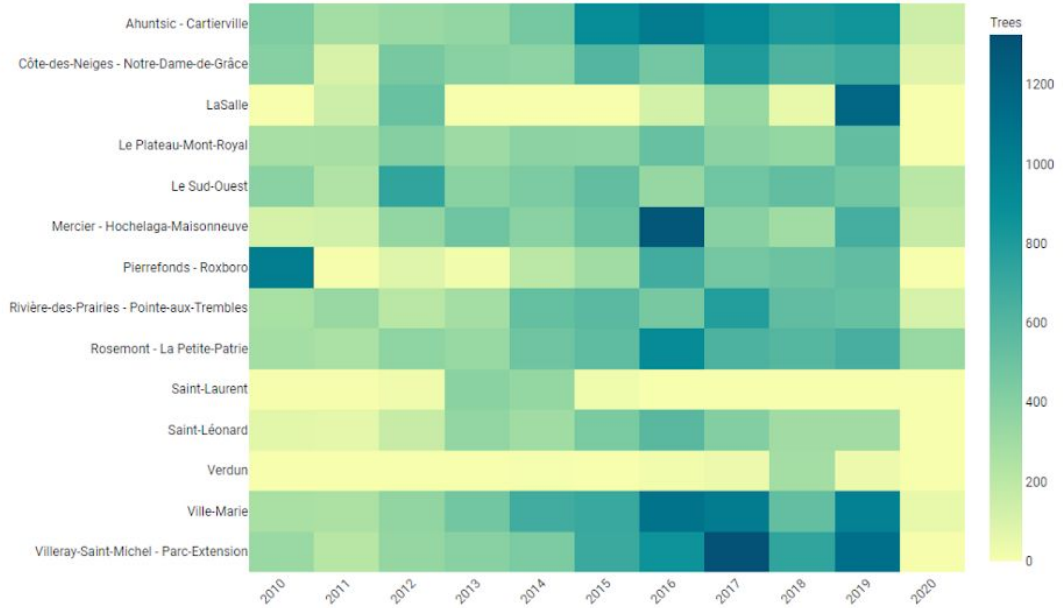
Date_Plantation	2010-12-31	2011-12-31	2012-12-31	2013-12-31	...	2017-12-31	2018-12-31	2019-12-31	2020-12-31
Arrond_Nom					...				
Ahuntsic - Cartierville	437.0	289.0	327.0	357.0	...	936.0	815.0	851.0	144.0
Côte-des-Neiges - Notre-Dame-de-Grâce	401.0	108.0	458.0	395.0	...	803.0	623.0	685.0	79.0
LaSalle	1.0	147.0	522.0	0.0	...	336.0	48.0	1175.0	0.0

	Date_Plantation	Counts
0	2017-05-16	4
1	2017-05-17	23
2	2017-05-18	2
3	2017-05-19	0
4	2017-05-20	0

Heatmap

Implement the main part of the data visualization

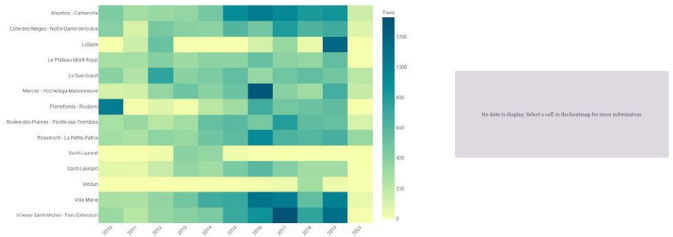
- Generate the heatmap from the preprocessed data and display - function ***'get_figure'***



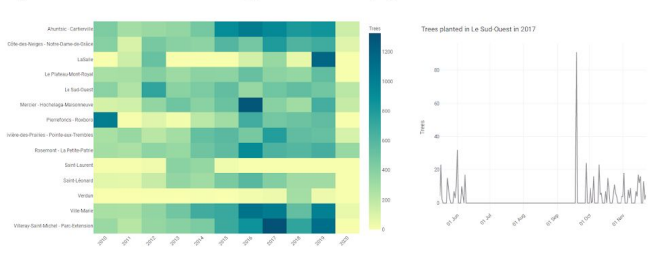
Line Chart

Build the line chart which accompanies the heatmap

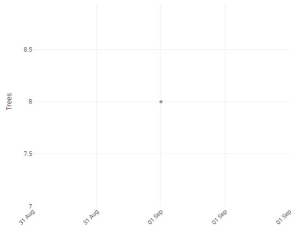
- Write the function to display an empty figure when there is no data to display - function ***'get_empty_figure'***
- Write the function to draw a grey rectangle on the empty figure when there is no data to display - function ***'add_rectangle_shape'***
- When there is data to display, generate the line chart with the corresponding data for the given neighborhood and year - function ***'get_figure'***



The heatmap with accompanying empty figure



The heatmap with accompanying line chart



The line chart when there is only data for one day to display

Theme et Tooltip

Theme

- Set it as the default theme, applied on top of the ***plotly_white***
- Complete the functions ***'create_custom_theme'*** et ***'set_default_theme'***.
- Make sure to closely follow the instructions in the comments of the code when defining each element of the theme as well as the images provided with the colors

Tooltip

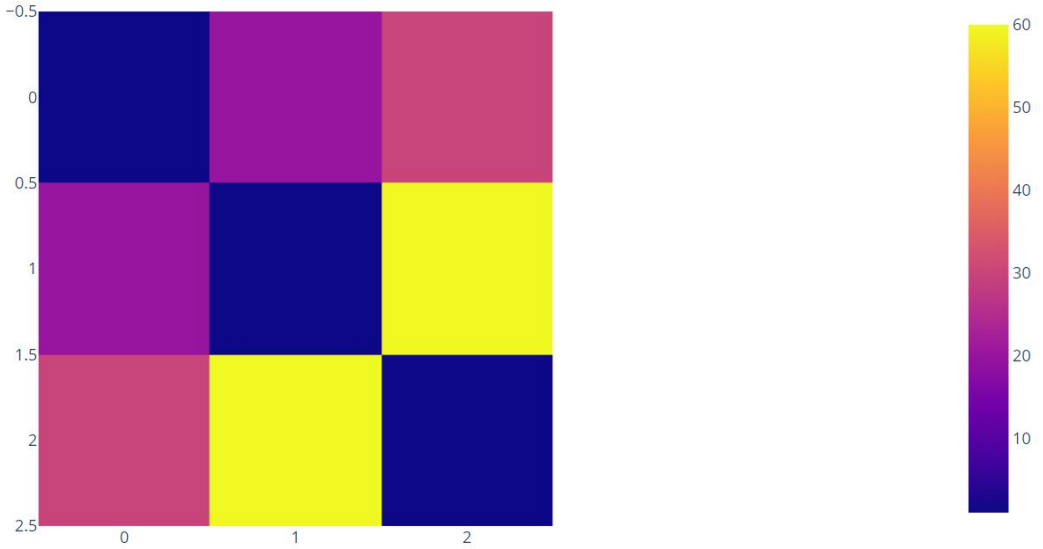
- the tooltip for a cell on heatmap should contain the name of the neighborhood, the year, and the amount of trees planted that year in the given neighborhood. In the line chart, the tooltip should contain the date and the amount of trees planted that day in the given neighborhood
- Complete the hover template for the heatmap - function ***'get_heatmap_hover_template'***
- Complete the hover template for the line chart - function ***'get_linechart_hover_template'***

Fonction ***imshow()***

- Plotly.express as px - *px.imshow()*
- Generate an image from a *numpy array* bidimensional
- One cell per each array element

```
import plotly.express as px

fig = px.imshow([[1, 20, 30],
                 [20, 1, 60],
                 [30, 60, 1]])
fig.show()
```



Heatmaps | Python | Plotly
plotly.express.imshow — 4.14.3 documentation

Fonction ***pd.Grouper()***

- A Grouper allows the user to specify a groupby instruction for an object.
- This specification will select a column via the key parameter, or if the level and/or axis parameters are given, a level of the index of the target object.
- If axis and/or level are passed as keywords to both Grouper and groupby, the values passed to Grouper take precedence.

```
>>> df = pd.DataFrame(
...     {
...         "Animal": ["Falcon", "Parrot", "Falcon", "Falcon", "Parrot"],
...         "Speed": [100, 5, 200, 300, 15],
...     }
... )
>>> df
   Animal  Speed
0  Falcon    100
1  Parrot      5
2  Falcon    200
3  Falcon    300
4  Parrot     15
>>> df.groupby(pd.Grouper(key="Animal")).mean()
        Speed
Animal
Falcon    200
Parrot     10
```

pandas.Grouper — pandas 1.2.2 documentation (pydata.org)

Submission

Deadline: May 29th 23h59