A brief introduction to Dash

INTERACTIVE DASHBOARD WITH PYTHON

GAËL PENESSOT | gael.penessot@data-decision.io | +33 677 773 537

What are we going to **learn** ?

- What is Dash ?
- How to install it
- How to start a Dash app
- How to display a figure (Plotly)
- How to create a dropdown menu
- How to make it interactive (callback functions)

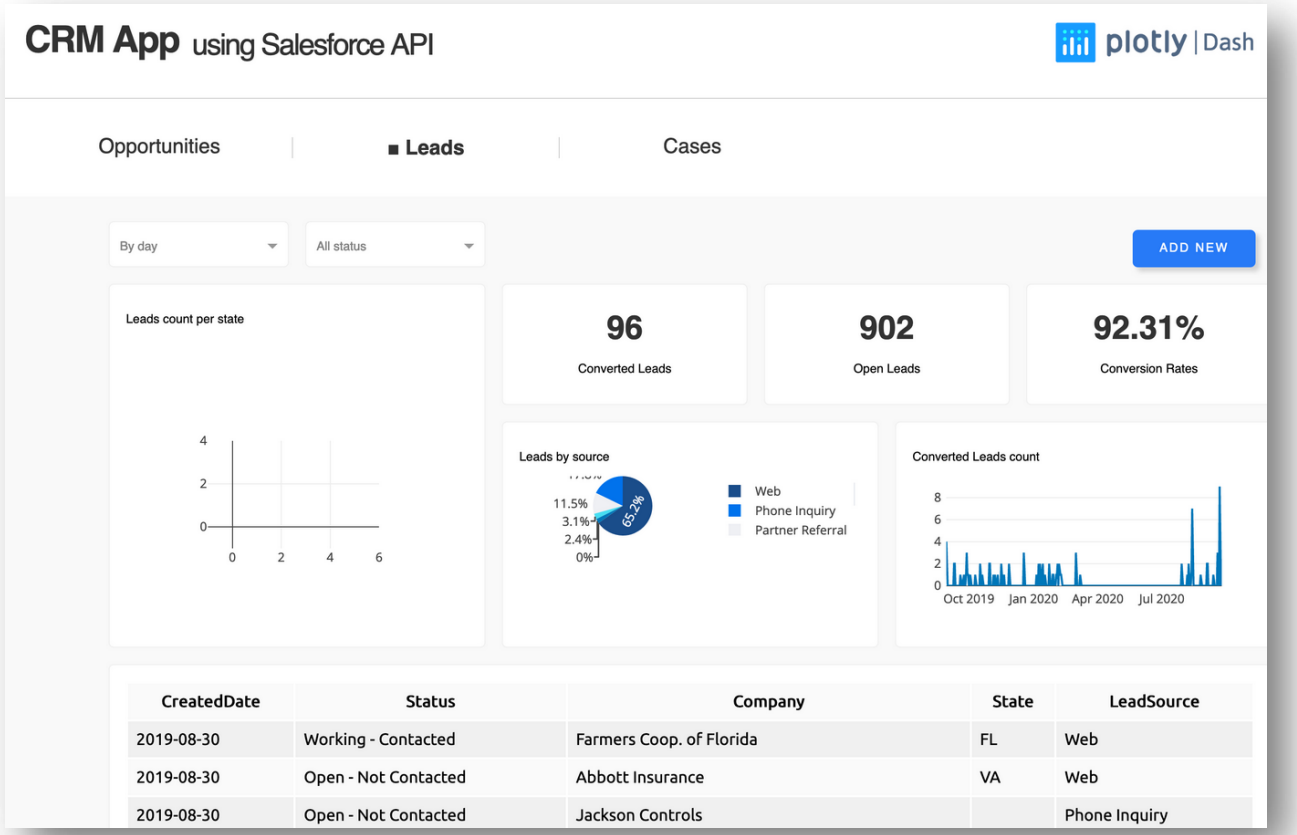
1. What is Dash ?

« Written on top of Plotly.js and React.js, Dash is ideal for building and deploying data apps with customized user interfaces in pure Python. It's particularly suited for anyone who works with data.

Through a couple of simple patterns, Dash abstracts away all of the technologies and protocols that are required to build a full-stack web app with interactive data visualization.

Dash is simple enough that you can bind a user interface to your Python code in less than 10 minutes.»

For those who knows **R**, Dash is the equivalent of **Shiny** or **flexdashboard**.



Documentation : link here

Gallery : link here

2. How to install Dash?

In your terminal, install dash.

```
pip install dash
```

This brings along three component libraries that make up the core of Dash:

- *dash_html_components*
- *dash_core_components*
- *dash_table*

And the *plotly* graphing library.

If you prefer Jupyter or JupyterLab as your development environment, I recommend installing jupyter-dash:

```
pip install jupyter-dash
```

3. How to start a app Dash? (1/5)

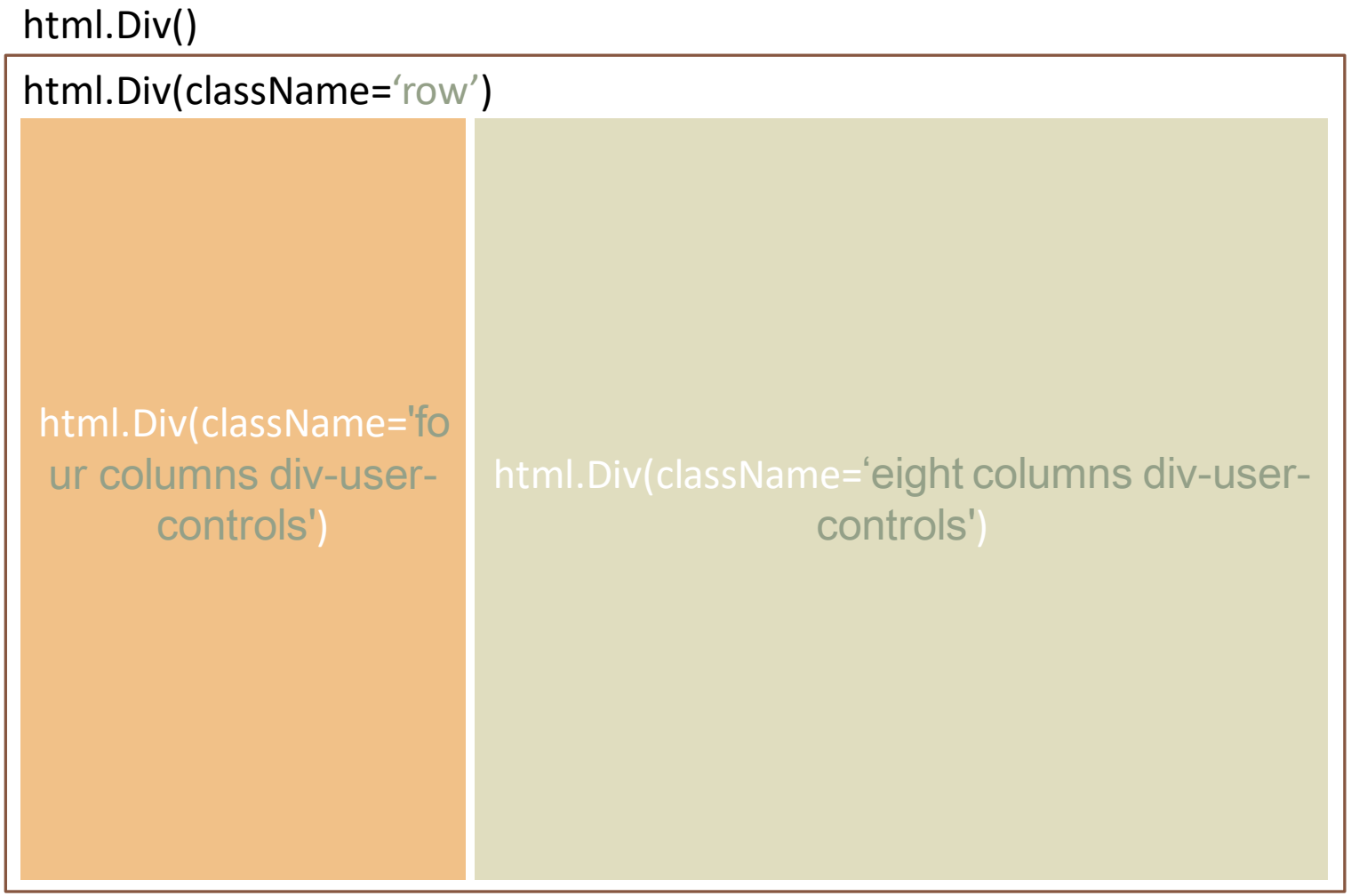
The following statements will load the necessary packages dash and dash_html_components. **Without any layout defined, the app won't start**. An empty html.Div is enough to get the app up and running.

```
import dash
import dash_html_components as html

# Initialise the app
app = dash.Dash(__name__)

# Define the app
app.layout = html.Div()

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

3. How to start a app Dash? (2/5)

Now, let's build a simple page with 2 columns !
The module *dash_html_components* provides you with several html components (Div, Img, etc.). The nesting of components is done via the **children** attribute.

The first html.Div() has one child. Another html.Div named row, which will contain all our content. The children of row are four columns div-user-controls and eight columns div-for-charts bg-grey



3. How to start a app Dash? (3/5)

Let’s code it!

```
import dash
import dash_html_components as html

# Initialise the app
app = dash.Dash(__name__)

# Define the app
app.layout = html.Div(children=[
                      html.Div(className='row',
                               children=[
                                  html.Div(className='four columns div-user-controls’),
                                  html.Div(className='eight columns div-for-charts bg-grey')
                                  ])
                                ])
# Run the app
if __name__ == ‘__main__’:
    app.run_server(debug=True)
```

3. How to start a app Dash? (4/5)

Now let’s first add some more information to our app, such as a title and a description.

For that, we use the Dash Components H2 to render a headline and P to generate html paragraphs.

```
children = [
            html.H2(‘2021 Gross Margin’),
            html.P(‘Visualizing time series with Plotly’),
            html.P(‘Pick one or more Manager to visualize results’)
]
```

3. How to start a app Dash? (5/5)

4. How to display a figure (Plotly) ?

First, let’s import some data :

```
import pandas as pd

df = pd.read_csv(‘2021_results_per_manager.csv’)
df.head()
```

	Manager	Date	Gross Margin (€)
0	FABIENNE.FAIVRE-JOLY	22/01/2021	4481,46
1	FABIENNE.FAIVRE-JOLY	03/02/2021	5162,47
2	FABIENNE.FAIVRE-JOLY	08/02/2021	620,14
3	FABIENNE.FAIVRE-JOLY	12/02/2021	12449,32
4	FABIENNE.FAIVRE-JOLY	14/02/2021	10,38

4. How to display a figure (Plotly) ?

Now, let's see how to create our first figure. With the building blocks for our web app in place, we can now define a plotly-graph. The function **dcc.Graph()** from dash_core_components uses the same figure argument as the plotly package.

```
import dash_core_components as dcc
import plotly.express as px
```

- Dash Core Components has a collection of useful and easy-to-use components, which add interactivity and functionalities to your dashboard.

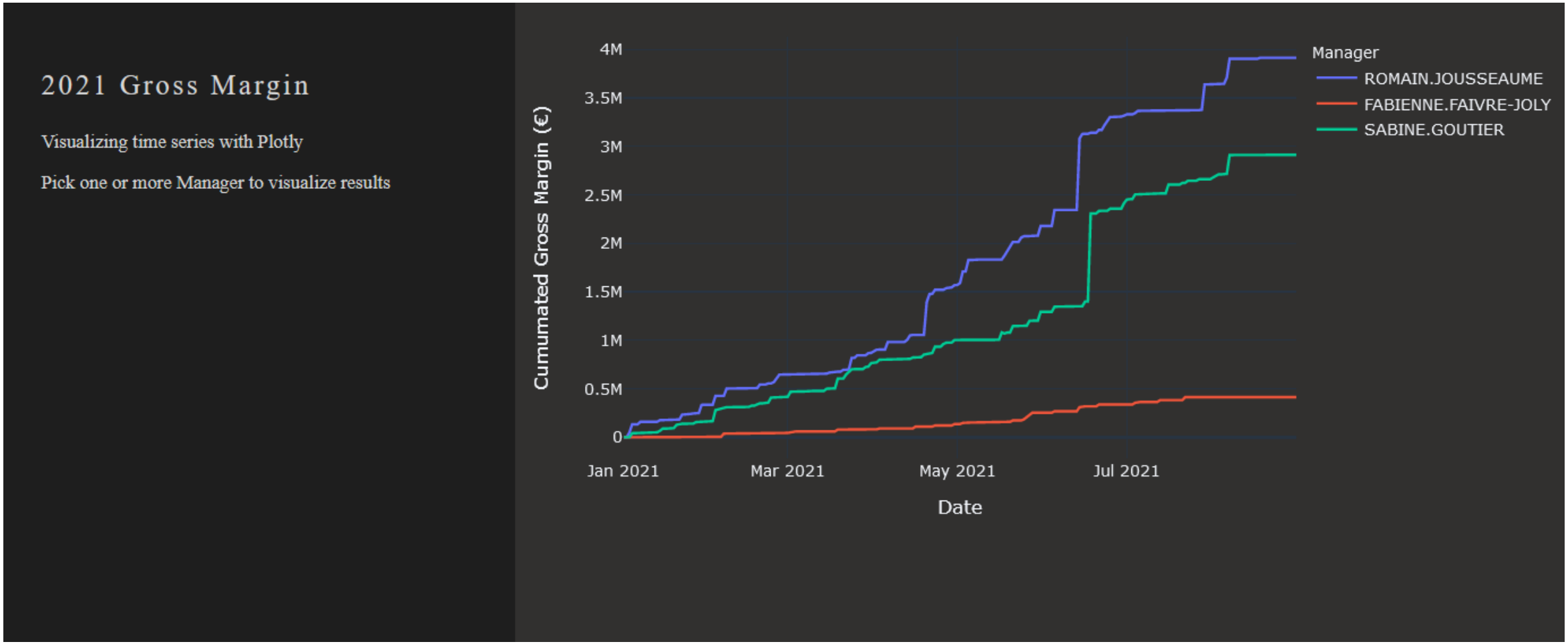
- Plotly Express is the express-version of plotly.py, which simplifies the creation of a plotly-graph, with the drawback of having fewer functionalities.

4. How to display a figure (Plotly) ?

Now let’s code it !

```
dcc.Graph(id=‘timeseries’,
          config={‘displayModeBar’: False},
          animate=True,
          figure=px.line(df,
                         x=‘Date’,
                         y=‘Cumulated Gross Margin (€)’,
                         color=‘Manager’,
                         template=‘plotly_dark').update_layout({
                              ‘plot_bgcolor’ : ‘rgba(0, 0, 0, 0)’,
                              ‘paper_bgcolor’ : ‘rgba(0, 0, 0, 0)’
                              })
           )
```

4. How to display a figure (Plotly) ?



4. How to create a dropdown menu?

Another core component is dcc.dropdown().

For our dropdown menu, we need a function that returns a list of dictionaries. The list contains dictionaries with two keys, label and value. The value of label is displayed in our app.

Add the following function to your script, before defining the app's layout.

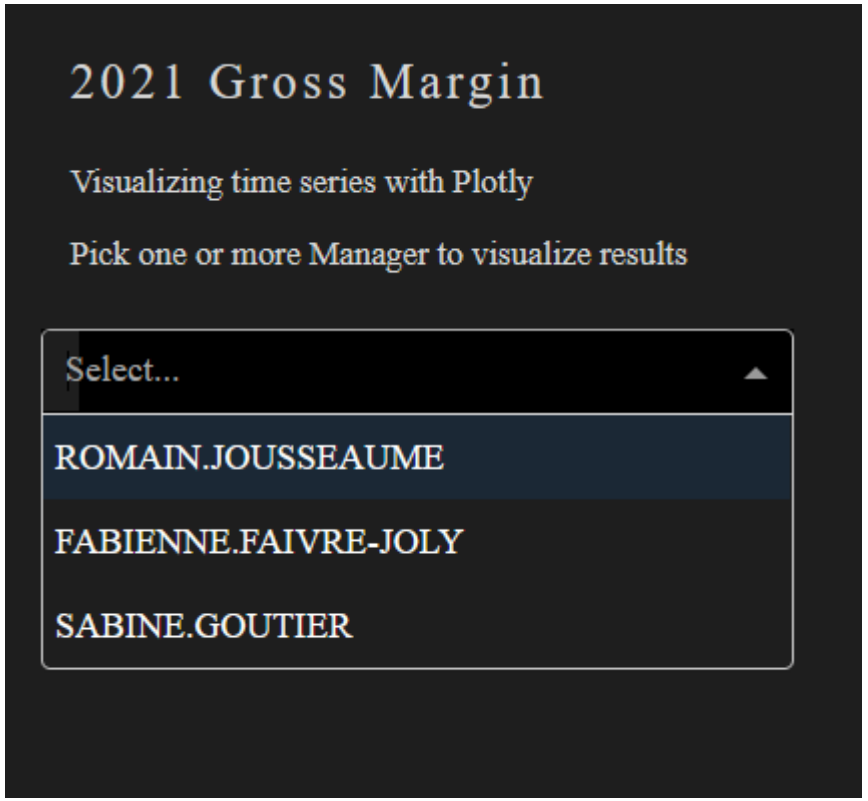
```
# Creates a list of dictionaries, which have the keys label and value.
def get_options(list_manager):
        dict_list = []
        for i in list_manager :
                dict_list.append({'label': i, 'value': i})
        return dict_list
```

4. How to create a dropdown menu?

We can now add dcc.Dropdown() to our app. Add a html.Div() as child to the list of children of four columns div-user-controls, with the argument className=div-for-dropdown. This html.Div() has one child, dcc.Dropdown().

```
html.Div(className='div-for-dropdown',
          children = [
               dcc.Dropdown(id='selector',
                            options=get_options(df['Manager'].unique()),
                            multi=True,
                            value=[df['Manager'].sort_values()[0]],
                            style={'backgroundColor': '#1E1E1E'},
                            className='selector') ],
                            style={'color': '#1E1E1E'})
```

4. How to display a figure (Plotly) ?



5. How to make it interactive (callback functions) ?

Callbacks add interactivity to your app. They can take inputs, for example, manager names selected via a dropdown menu, pass these inputs to a function and pass the return value of the function to another component.

A callback will pass the selected values from the dropdown to the function and return the figure to a dcc.Graph() in our app.

Multiple inputs and outputs are possible, but for now, we will start with a single input and a single output. We need the class *dash.dependencies.Input* and *dash.dependencies.Output*.

Add the following line to your import statements:

```
from dash.dependencies import Input, Output
```

5. How to make it interactive (callback functions) ?

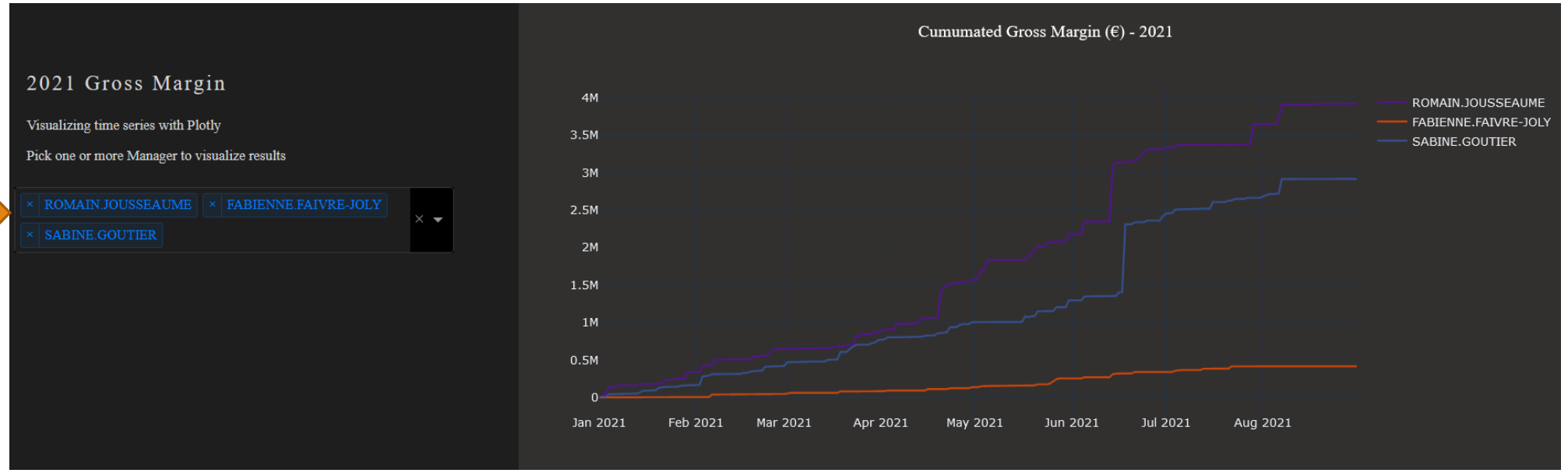
The function draws the traces of a plotly-figure based on the stocks which were passed as arguments and returns a figure that can be used by dcc.Graph().

The inputs for our function are given in the order in which they were set in the callback. Names chosen for the function's arguments do not impact the way values are assigned.

```
@app.callback(Output('timeseries', 'figure'),
              [Input('selector', 'value')])
def update_timeseries(selected_dropdown_value):
        trace = []
        df_sub = df
        for manager in selected_dropdown_value:
                trace.append(
                        go.Scatter(x=df_sub[df_sub['Manager'] == manager].index,
                                   y=df_sub[df_sub['Manager'] == manager]['value'],
                                   mode='lines',
                                   opacity=0.7,
                                   name= manager,
                                   textposition='bottom center'))
        traces = [trace]
        data = [val for sublist in traces for val in sublist]
        figure = {'data': data,
                  'layout': go.Layout(colorway=["#5E0DAC", '#FF4F00', '#375CB1',
'#FF7400', '#FFF400', '#FF0056'],
                  template='plotly_dark',
                  paper_bgcolor='rgba(0, 0, 0, 0)',
                  plot_bgcolor='rgba(0, 0, 0, 0)',
                  margin={'b': 15}, hovermode='x',
                  autosize=True,
                  title={'text': 'Stock Prices', 'font': {'color': 'white'}, 'x': 0.5},
                                   xaxis={'range': [df_sub.index.min(),
df_sub.index.max()]}, ), }
        return figure
```

5. How to make it interactive (callback functions) ?

3 managers selected



2 managers selected