

DATA- CHALLENGE

COVID-19 en UE

with

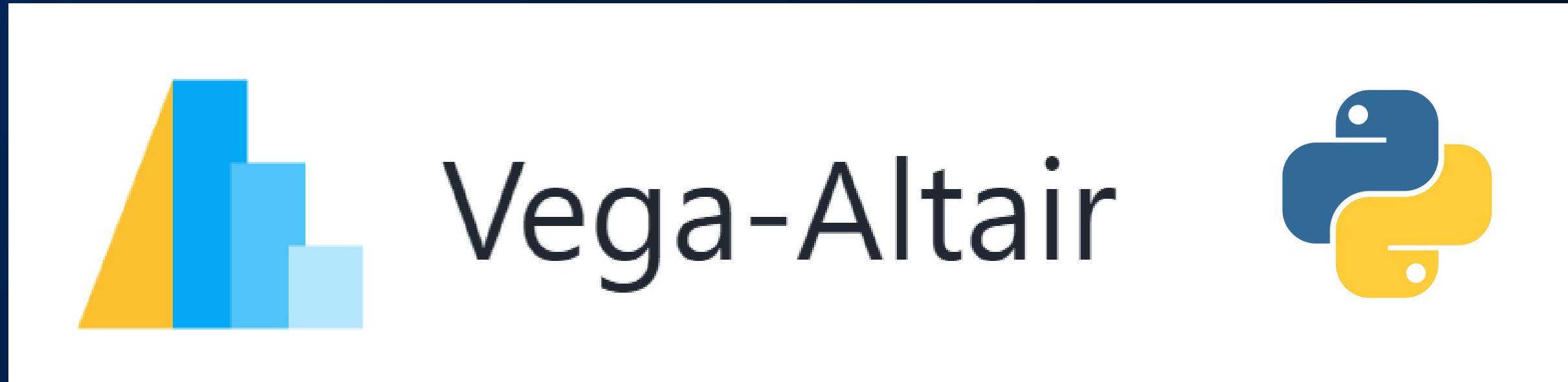
PALOMBIER WILLIAM
KAYGUN ENIS
BEN AMMAR AZIZ



16/12/2024



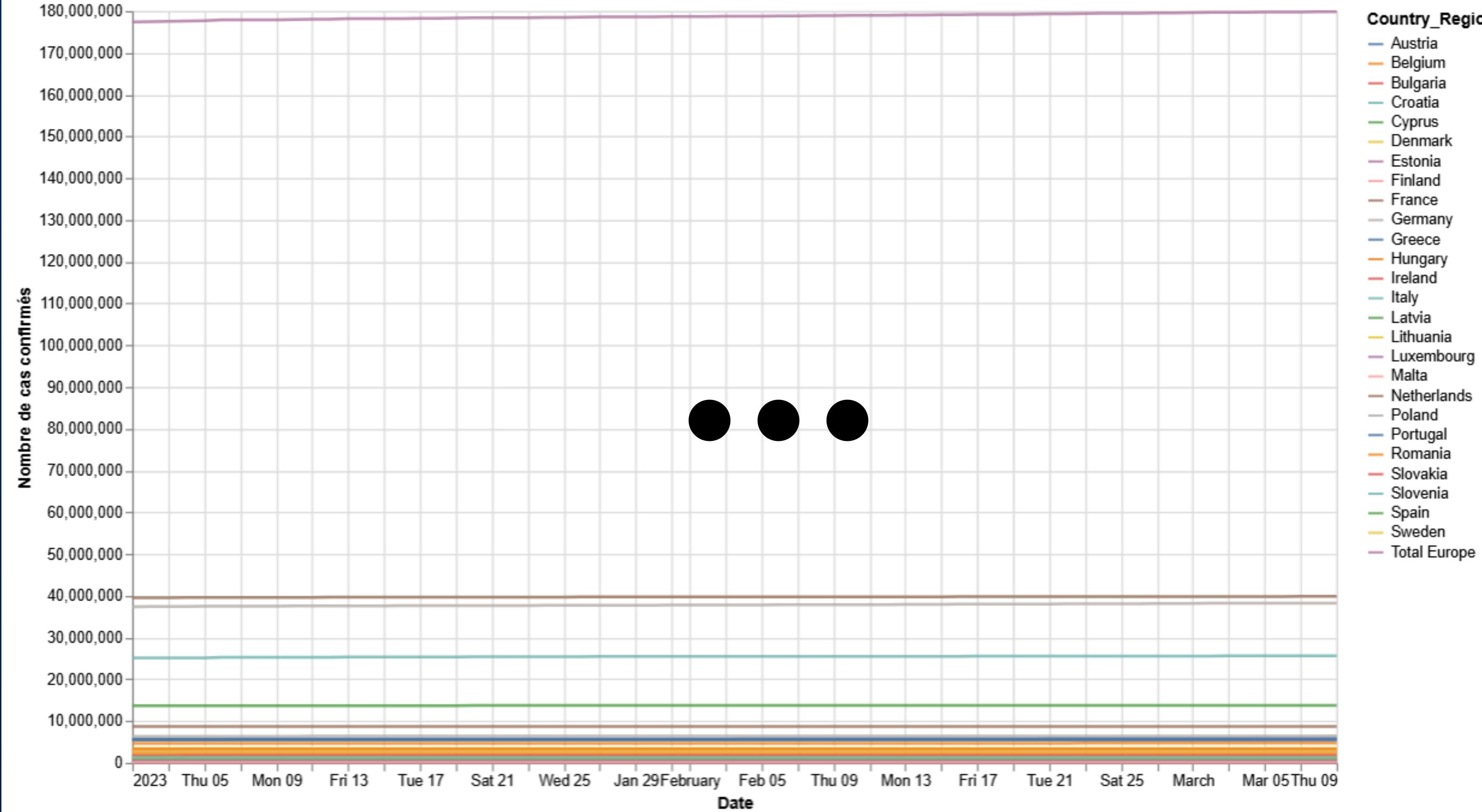
Package utilisé pour le graphique (sur python) :



Graphique dynamique interactif

Évolution des cas COVID-19 dans l'Union Européenne

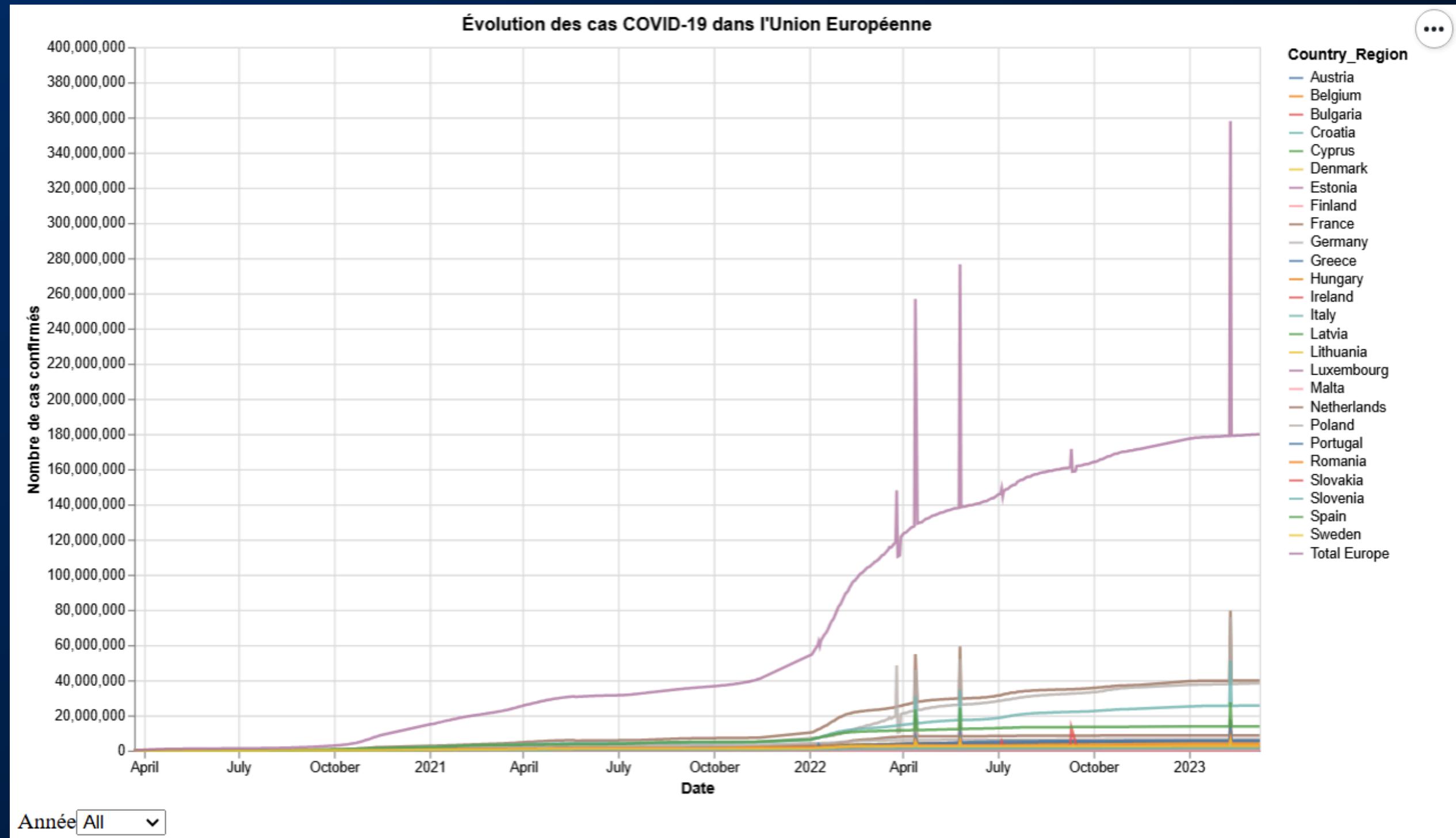
...



Uniformiser les données !

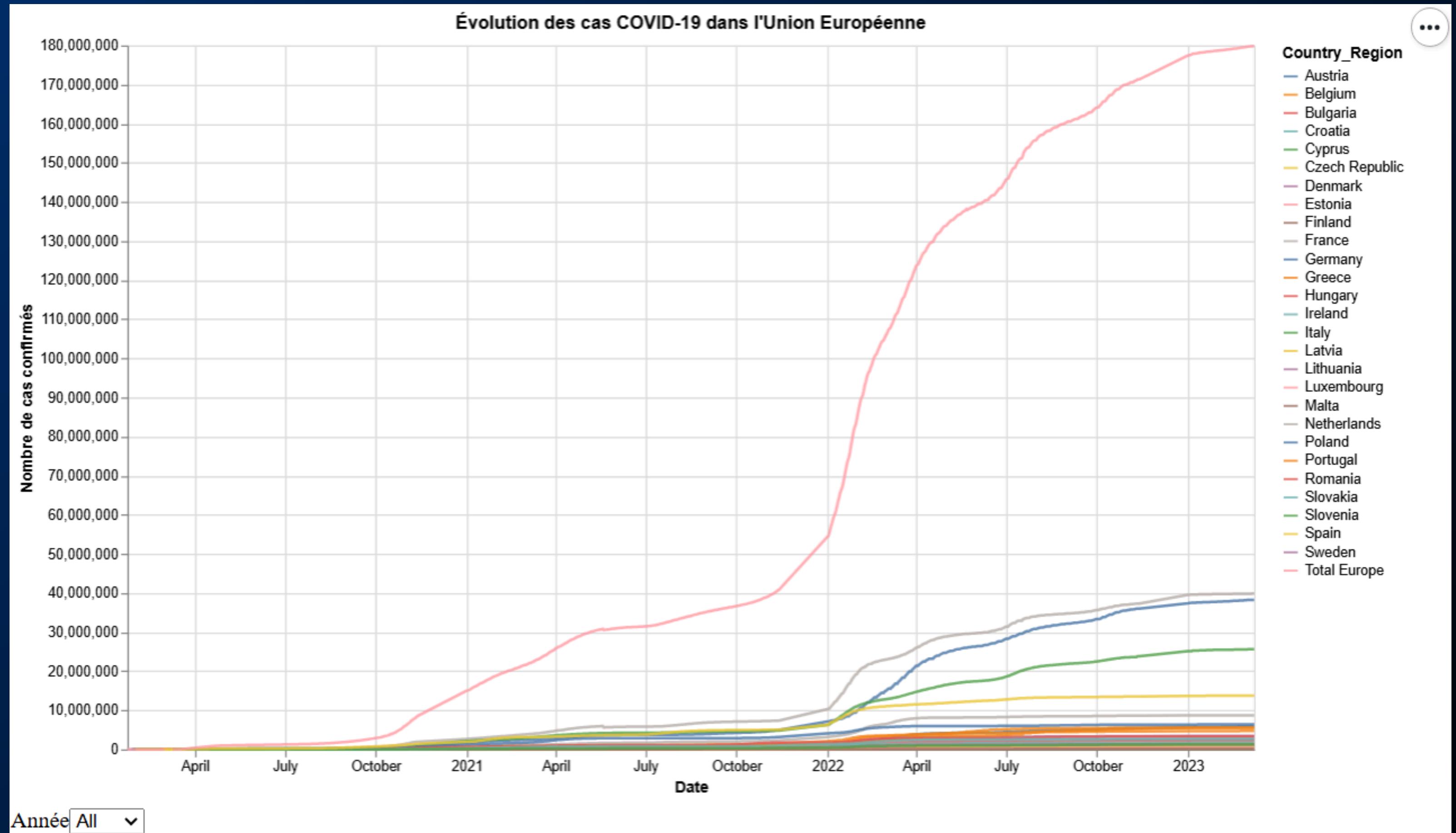
```
13         df = pd.read_csv(file_path)
14         df = df.rename(columns={
15             'Country/Region': 'Country_Region',
16             'Last Update': 'Last_Update',
17             'Province/State': 'Province_State',
18             'Latitude': 'Lat',
19             'Longitude': 'Long_'
20         })
```

Des valeurs doublées...

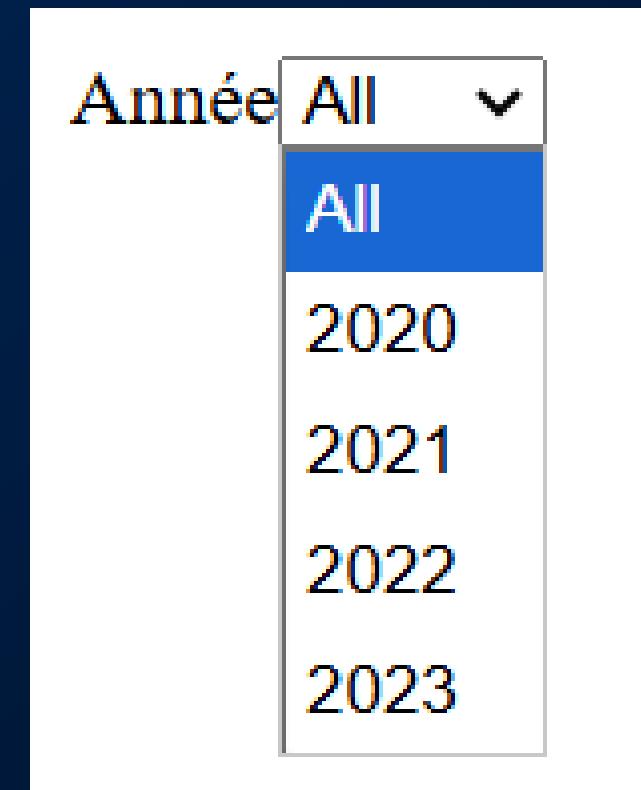


```
file_date = datetime.strptime(filename.split('.')[0], '%m-%d-%Y').date()  
df['Date'] = pd.to_datetime(file_date)
```

Un graphique fonctionnel

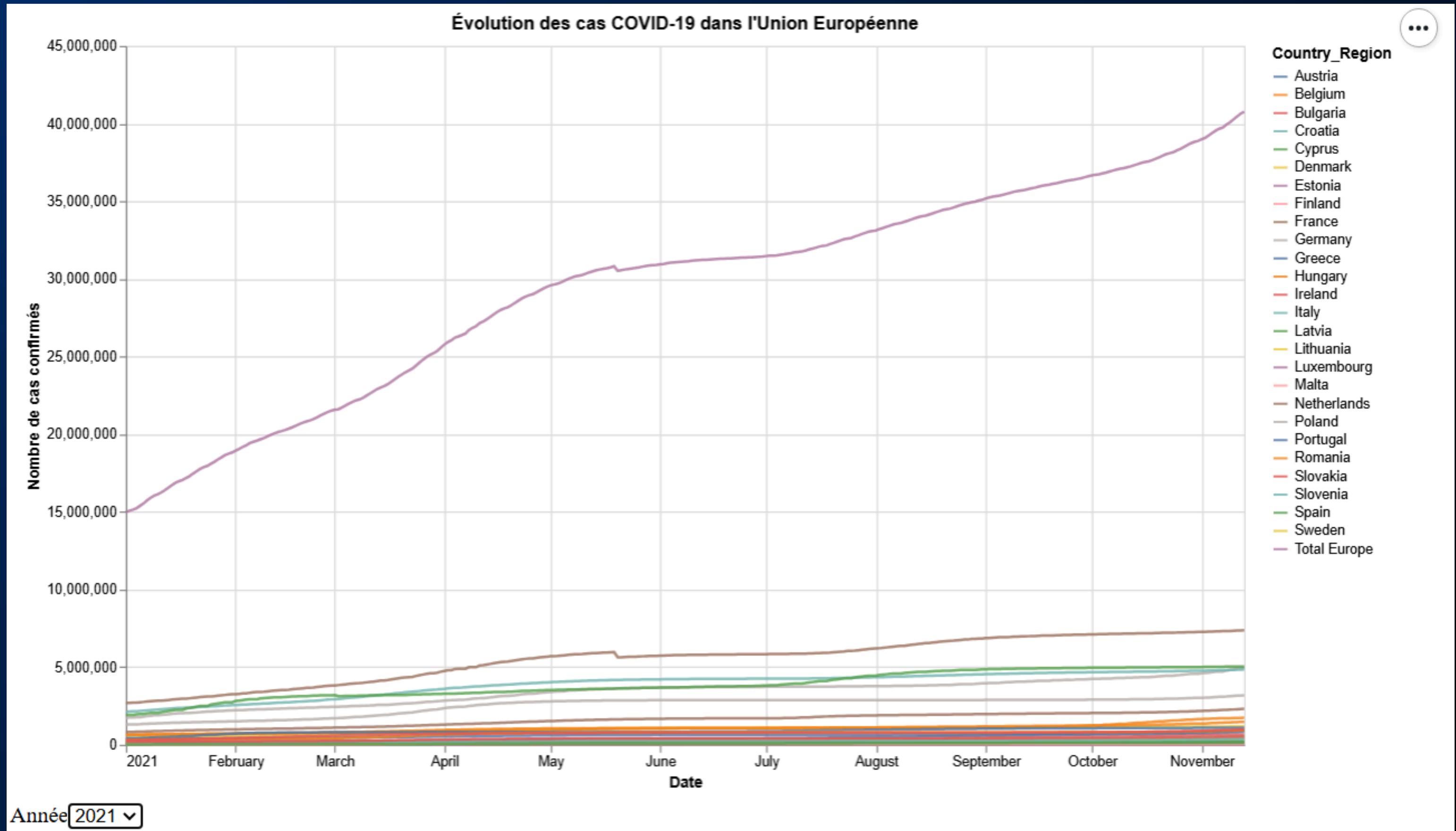


Menu déroulant

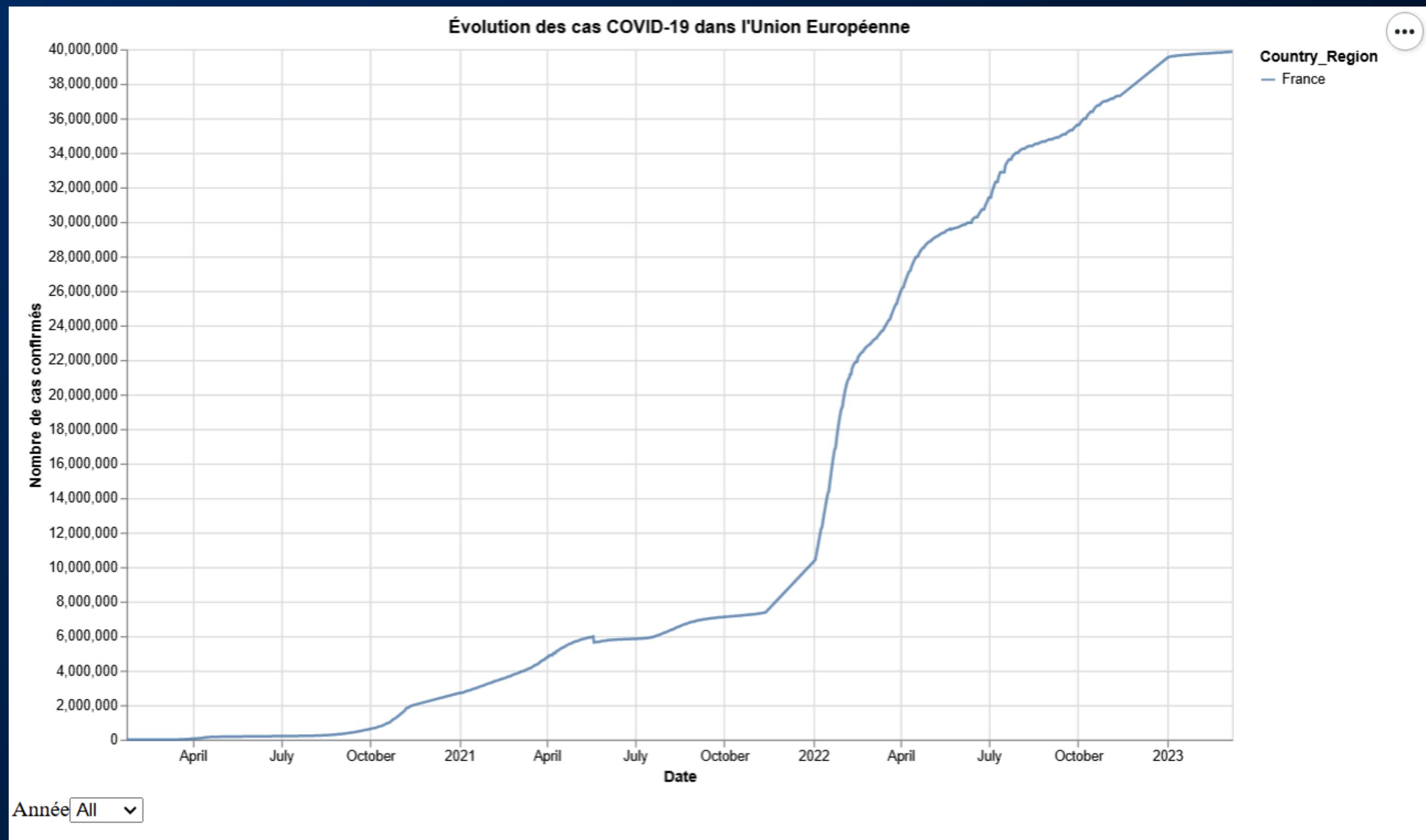


```
year_options = ['All'] + [str(y) for y in sorted(combined_data['Année'].unique())]
year_param = alt.param(
    name='Année',
    bind=alt.binding_select(options=year_options, name='Année'),
    value='All'
)
```

Pour l'année 2021



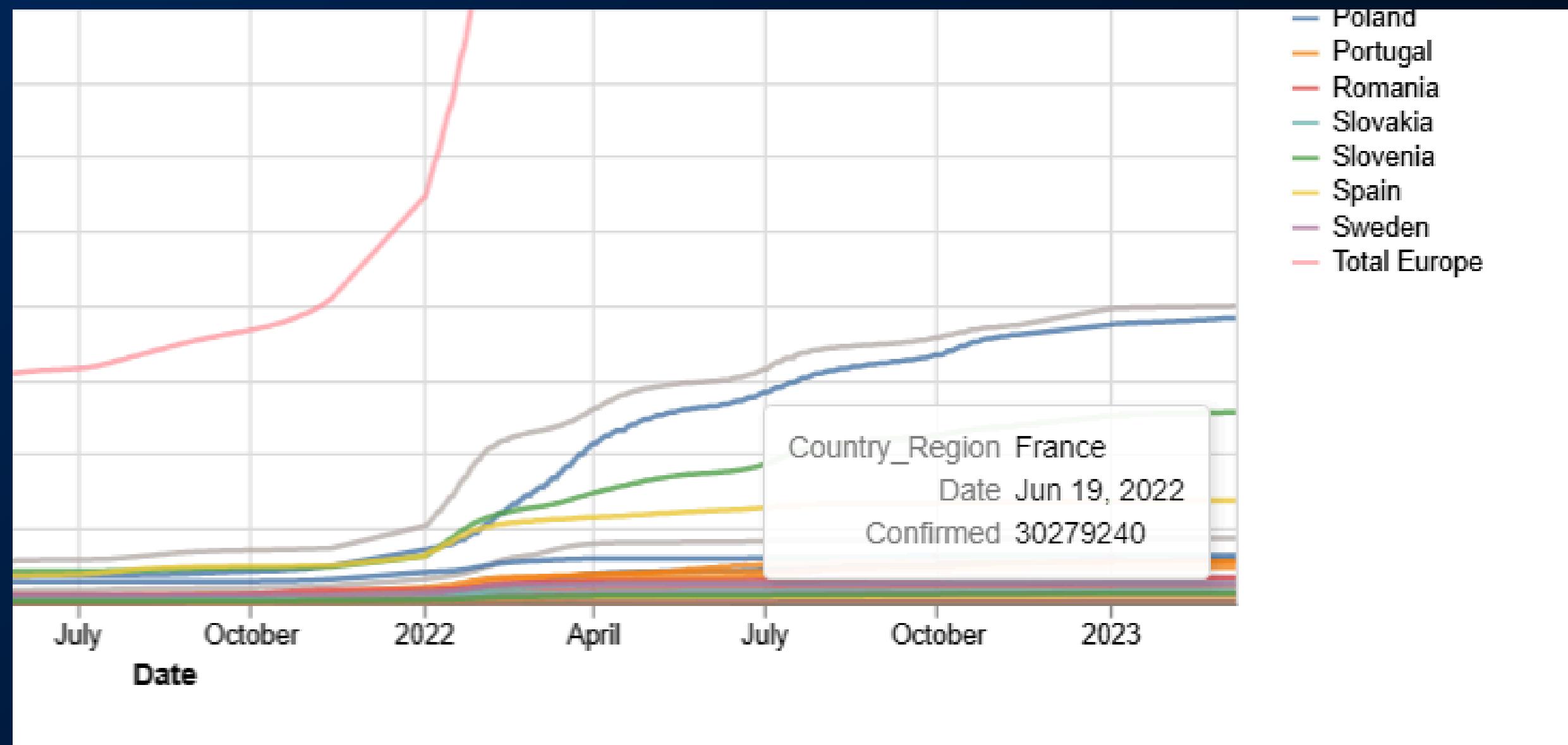
Légende interactive



```
70     selection = alt.selection_point(fields=[ 'Country_Region' ], bind='legend')
```

Ajout d'une infobulle

```
confirmed_line = base.mark_line(opacity=0.8).encode(  
    tooltip=['Country_Region', 'Date', 'Confirmed'])  
)
```



Carte interactive

Packages utilisés :

`glob` : pour gérer les fichiers dans un dossier.

`pandas` : pour manipuler les données.

`plotly.express` : pour créer des visualisations interactives.

`numpy` : pour effectuer des opérations mathématiques.

`plotly.io` : pour exporter le résultat en HTML.



Nettoyage des données

Fonction clean_data:

Cette fonction s'assure que les données sont fiables et exploitables pour la carte. Voici les étapes principales :

- **Détection des valeurs aberrantes :**

Les seuils maximums sont définis pour les colonnes **Confirmed** (10 milliards), **Deaths** (100 millions), et **Recovered** (10 milliards). Toute valeur au-delà de ces seuils est remplacée par **NaN** (valeur manquante).

Exemple : Si un pays affiche 50 milliards de cas confirmés, cette valeur sera exclue.

- **Remplacement des valeurs manquantes :**

Pour **Confirmed** et **Deaths**, les valeurs manquantes sont remplacées par la **médiane** de leur colonne respective.

Pour **Recovered**, les valeurs manquantes sont remplacées par **0**, car ce choix reflète mieux une absence de données sur les guérisons.

```
def clean_data(df):
    max_cases = 10_000_000_000
    max_deaths = 100_000_000
    max_recoveries = 10_000_000_000

    df['Confirmed'] = np.where(df['Confirmed'] > max_cases, np.nan, df['Confirmed'])
    df['Deaths'] = np.where(df['Deaths'] > max_deaths, np.nan, df['Deaths'])
    df['Recovered'] = np.where(df['Recovered'] > max_recoveries, np.nan, df['Recovered'])

    df['Confirmed'] = df['Confirmed'].fillna(df['Confirmed'].median())
    df['Deaths'] = df['Deaths'].fillna(df['Deaths'].median())
    df['Recovered'] = df['Recovered'].fillna(0)
```

Création de la carte

Fonction `create_time_series_covid_map`:

Le script filtre les données pour se concentrer uniquement sur les pays européens.

Les données sont agrégées par date et par pays pour obtenir les totaux quotidiens de cas confirmés, décès et guérisons.

Une carte **choroplèthe** est créée avec :

- Les couleurs représentant les cas confirmés.
- Une animation permettant de visualiser l'évolution dans le temps.
- Un texte personnalisé dans les infobulles, notamment pour les cas de guérison manquants.

```
df['Last_Update'] = pd.to_datetime(df['Last_Update'], format='mixed')
df['Last_Update'] = df['Last_Update'].dt.date
df['Country_Region'] = df['Country_Region'].str.strip().str.title()

df_eu = df[df['Country_Region'].isin(eu_countries)]

daily_data = df_eu.groupby([
    'Last_Update',
    'Country_Region'
]).agg({
    'Confirmed': 'sum',
    'Deaths': 'sum',
    'Recovered': 'sum',
    'Lat': 'first',
    'Long_': 'first'
}).reset_index()

fig = px.choropleth(
    daily_data,
    locations='Country_Region',
    locationmode='country names',
    color='Confirmed',
    hover_name='Country_Region',
    color_continuous_scale='Viridis',
    animation_frame='Last_Update',
    title='COVID-19 Daily Confirmed Cases in EU Countries',
    labels={
        'Confirmed': 'Confirmed Cases',
        'Deaths': 'Total Deaths',
        'Recovered_Display': 'Recovered Cases',
        'Last_Update': 'Date'
    },
    hover_data={
        'Confirmed': True,
        'Deaths': True,
        'Recovered': False,
        'Recovered_Display': True,
        'Last_Update': True
    }
)

fig.update_layout(
    title_x=0.5,
    geo=dict(
        showframe=False,
        showcoastlines=True,
        projection_type='mercator',
        scope='europe',
        lonaxis=dict(range=[-30, 60]),
        lataxis=dict(range=[30, 75])
    )
)

return fig, daily_data
```

Chargement des fichiers de données

Tous les fichiers CSV dans le dossier spécifié sont lus et combinés en un seul grand tableau. Ensuite, les données sont nettoyées grâce à la fonction **clean_data**.

```
# Load data
path = r'C:\Users\Admin\Desktop\M1 IDS\Data chall\covid_data'
filenames = glob.glob(path + "/*.csv")

dfs = [pd.read_csv(filename, sep=',') for filename in filenames]
big_frame = pd.concat(dfs, ignore_index=True)

big_frame = clean_data(big_frame)

covid_time_series_map, daily_data = create_time_series_covid_map(big_frame)
```

Exportation de la carte

La carte interactive est enregistrée au format **HTML** avec **pio.write_html**, permettant de la consulter **facilement** dans un navigateur.

```
# Download HTML
pio.write_html(covid_time_series_map, file='eu_covid_map.html', auto_open=False)

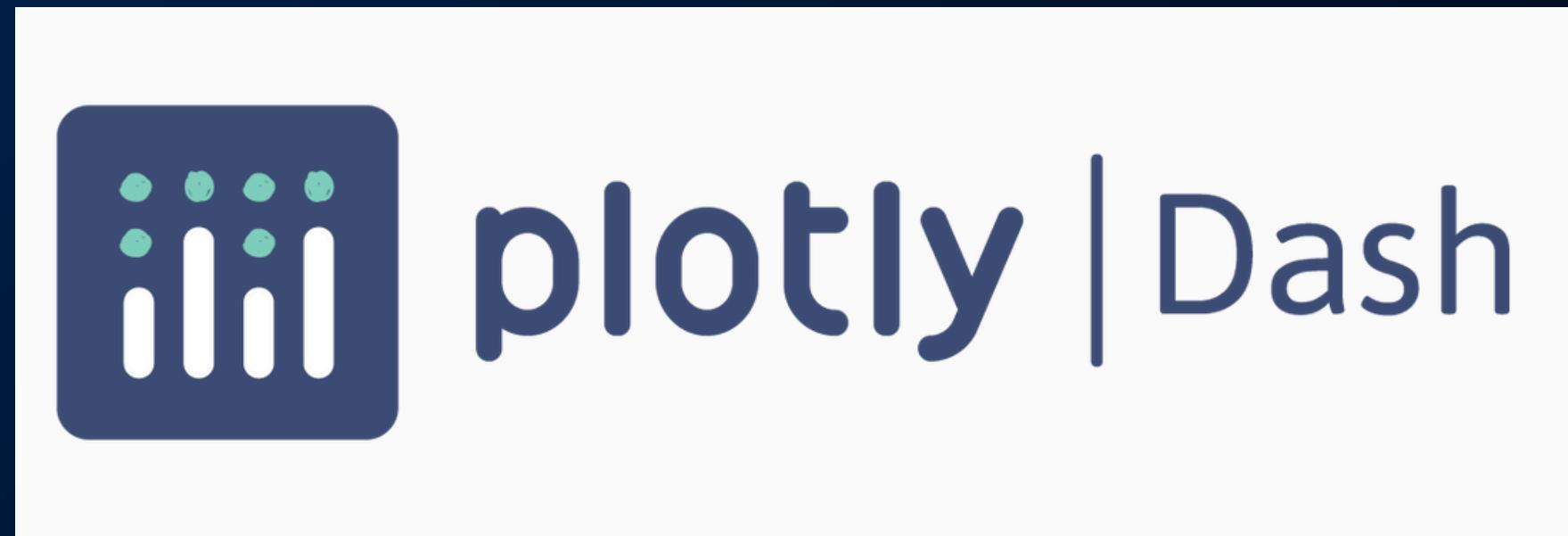
print("HTML map downloaded as eu_covid_map.html")
```

Dashboard Intéractif

Packages utilisés:

Plotly express: interface haut-niveau de la bibliothèque Plotly , simplifiant la création de graphiques interactifs.

Dash: framework Python basé sur Flask, Plotly et React.js, permettant de construire des applications web analytiques interactives sans avoir besoin de maîtriser HTML, CSS ou JavaScript.



Initialisation de l'application

Initialise une application Dash et définit le titre de la page web (visible dans l'onglet du navigateur).

```
app = dash.Dash(__name__)
app.title = "Tableau de bord COVID-19 - Top 10 UE"
```

Mise en page de l'application

Configure la structure visuelle de l'application:

html.H1 : Titre principal.

html.Div : Conteneurs principaux pour organiser les composants.

dcc.Slider : Curseur pour sélectionner une date.

dcc.Graph : Composants pour afficher les graphiques (carte, barplot, camembert).

Applique un **thème sombre** (backgroundColor: '#111111').

```
app.layout = html.Div(style={'backgroundColor': dark_theme['background'], 'padding': '20px'}, children=[  
    html.H1("Tableau de bord COVID-19 - Top 10 Pays de l'UE",  
           style={'textAlign': 'center', 'color': dark_theme['text'], 'fontSize': '36px'}),  
    html.Div([  
        html.Label("Sélectionnez une date :", style={'color': dark_theme['text'], 'fontSize': '18px'}),  
        dcc.Slider(  
            id='date-slider',  
            min=0,  
            max=len(available_dates) - 1,  
            value=len(available_dates) - 1,  
            marks={i: str(pd.to_datetime(date).date()) for i, date in enumerate(available_dates)},  
            step=1  
        ),  
        ], style={'marginBottom': '30px'}),  
    html.Div([  
        dcc.Graph(id='choropleth-map', style={'height': '600px', 'width': '100%'})  
    ], style={'marginBottom': '30px'}),  
    html.Div([  
        html.Div([  
            dcc.Graph(id='bar-chart', style={'height': '500px'})  
        ], style={'width': '48%', 'display': 'inline-block', 'padding': '10px'}),  
        html.Div([  
            dcc.Graph(id='pie-chart', style={'height': '500px'})  
        ], style={'width': '48%', 'display': 'inline-block', 'padding': '10px'}),  
    ]),  
    html.Div(id='data-table', style={'overflowX': 'auto', 'paddingTop': '20px'})  
])
```

Callback pour la carte choroplète, les graphiques et les données

Gérer les interactions entre l'utilisateur et les éléments visuels du tableau de bord.

Entrées (Input)

Permet de fournir l'index d'une date sélectionnée par l'utilisateur dans la liste des dates disponibles avec l'ID date-slider.

Sorties (Outputs)

Carte choroplète (figure) : Affiche les cas confirmés sur la carte en fonction des données filtrées.

Graphique à barres (figure) : Affiche les cas confirmés sous forme de barplot pour les 10 pays les plus touchés.

Camembert (figure) : Visualise la répartition des cas confirmés sous forme de pourcentages.

Tableau de données (children) : Génère un tableau HTML pour afficher les informations détaillées des pays (classement, pays, cas confirmés).

```
# Callback pour La carte choroplète, les graphiques et les données
@app.callback(
    [Output('choropleth-map', 'figure'),
     Output('bar-chart', 'figure'),
     Output('pie-chart', 'figure'),
     Output('data-table', 'children')],
    [Input('date-slider', 'value')]
)

def update_dashboard(selected_date_index):
    # Convertir l'index du curseur en date réelle
    selected_date = available_dates[selected_date_index]
    filtered_df = df[df['Date'] == selected_date]

    # Classement des pays (Top 10 uniquement, sans doublons)
    filtered_df = filtered_df.groupby('Country/Region', as_index=False).max()
    filtered_df = filtered_df.nlargest(10, 'Confirmed')
    filtered_df['Rank'] = range(1, len(filtered_df) + 1)
```

```
# Carte choroplète
choropleth_fig = px.choropleth(
    filtered_df,
    locations='Country/Region',
    locationmode='country names',
    color='Confirmed',
    title=f"Top 10 des cas confirmés au {pd.to_datetime(selected_date).date()}",
    labels={'Confirmed': 'Cas Confirmés'},
    color_continuous_scale='YlOrRd',
    template='plotly_dark'
)
choropleth_fig.update_geos(
    showland=True, landcolor="#0d0d0d",
    showocean=True, oceancolor="#1e1e1e",
    showcountries=True, countrycolor=dark_theme['grid'],
    projection_type="natural earth",
    center={"lat": 50, "lon": 10},
    lataxis_range=[35, 65],
    lonaxis_range=[-25, 40]
)
choropleth_fig.update_layout(
    margin=dict(l=0, r=0, t=50, b=0),
    font=dict(color=dark_theme['text'])
)
```

Génère une carte interactive affichant les pays, colorés en fonction du nombre de cas confirmés avec un titre dynamique affichant la date sélectionnée.

```
# Graphique à barres
bar_chart_fig = px.bar(
    filtered_df,
    x='Country/Region',
    y='Confirmed',
    title=f"Distribution des cas confirmés au {pd.to_datetime(selected_date).date()}",
    labels={'Confirmed': 'Cas Confirmés', 'Country/Region': 'Pays'},
    template='plotly_dark',
    color='Country/Region'
)
bar_chart_fig.update_layout(
    margin=dict(l=0, r=0, t=50, b=50),
    font=dict(color=dark_theme['text'])
)
```

Crée un barplot comparant les cas confirmés des 10 pays les plus touchés.

```
# Camembert
pie_chart_fig = px.pie(
    filtered_df,
    values='Confirmed',
    names='Country/Region',
    title=f"Part des cas confirmés au {pd.to_datetime(selected_date).date()}",
    template='plotly_dark'
)
pie_chart_fig.update_layout(
    margin=dict(l=0, r=0, t=50, b=50),
    font=dict(color=dark_theme['text'])
)
```

Affiche la proportion des cas confirmés pour chaque pays dans le top 10.

```

# Tableau des données
table = filtered_df[['Rank', 'Country/Region', 'Confirmed']].sort_values(by='Rank')
table_html = html.Table(
    style={'width': '100%', 'color': dark_theme['text'], 'border': f"1px solid {dark_theme['grid']}"},
    children=[
        html.Thead(html.Tr([html.Th(col) for col in table.columns])),
        html.Tbody([
            html.Tr([html.Td(table.iloc[i][col]) for col in table.columns]) for i in range(len(table))
        ])
    ]
)

return choropleth_fig, bar_chart_fig, pie_chart_fig, table_html

```

Génère un tableau HTML avec :

- Rang : Classement des pays.
- Nom du pays.
- Nombre de cas confirmés.

Retour du callback

Retourne les objets graphiques et HTML à insérer dans leurs composants respectifs:

- La carte choroplète dans choropleth-map.
- Le barplot dans bar-chart.
- Le camembert dans pie-chart.
- Le tableau dans data-table.

```
if __name__ == '__main__':
    app.run_server(debug=True, port=8051)
```

Démarre le serveur Dash pour rendre l'application accessible via un navigateur web.

`debug=True` : Recharge automatiquement le serveur lors des modifications du code.

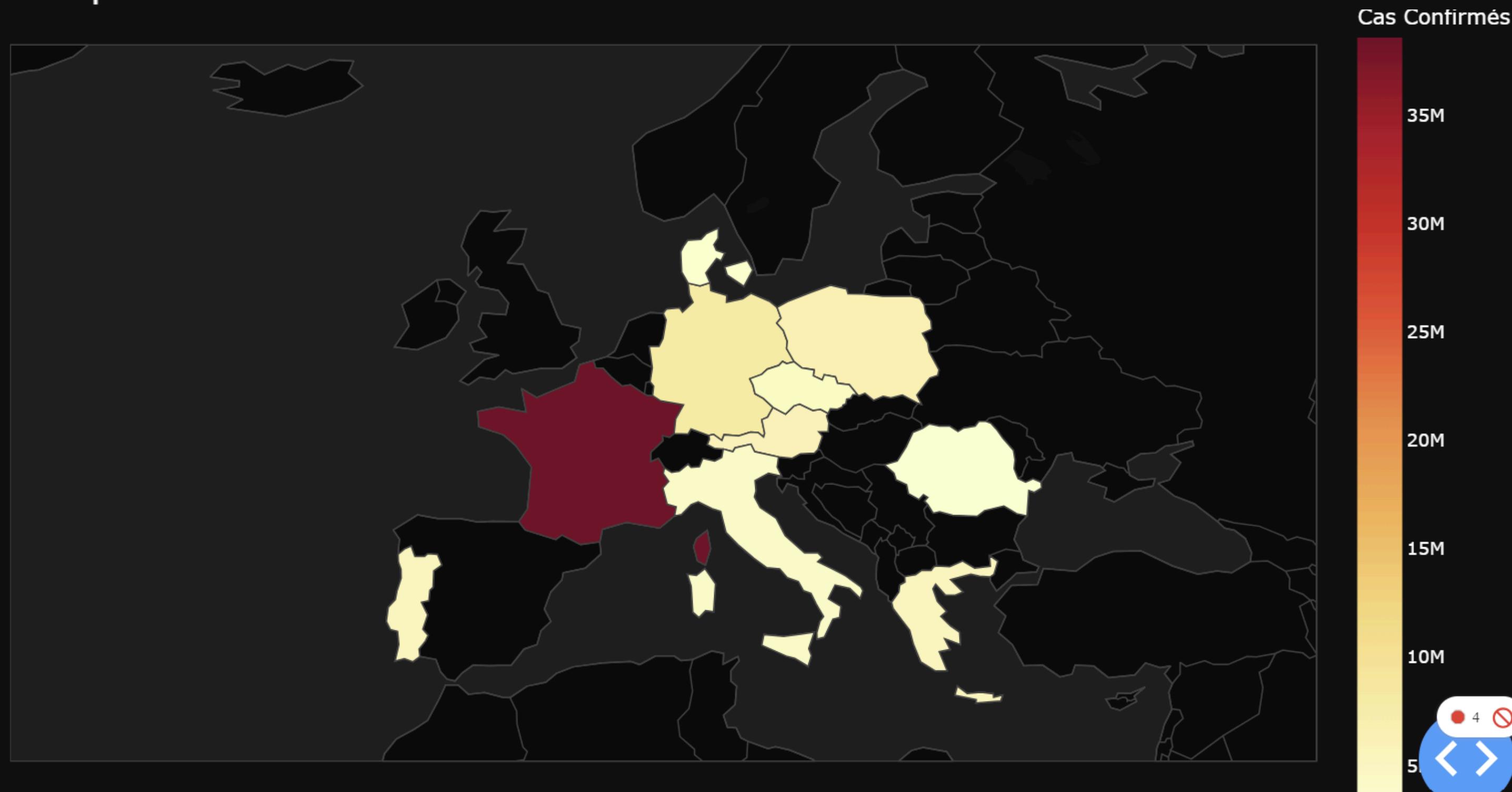
`port=8051` : Spécifie le port sur lequel l'application écoute.

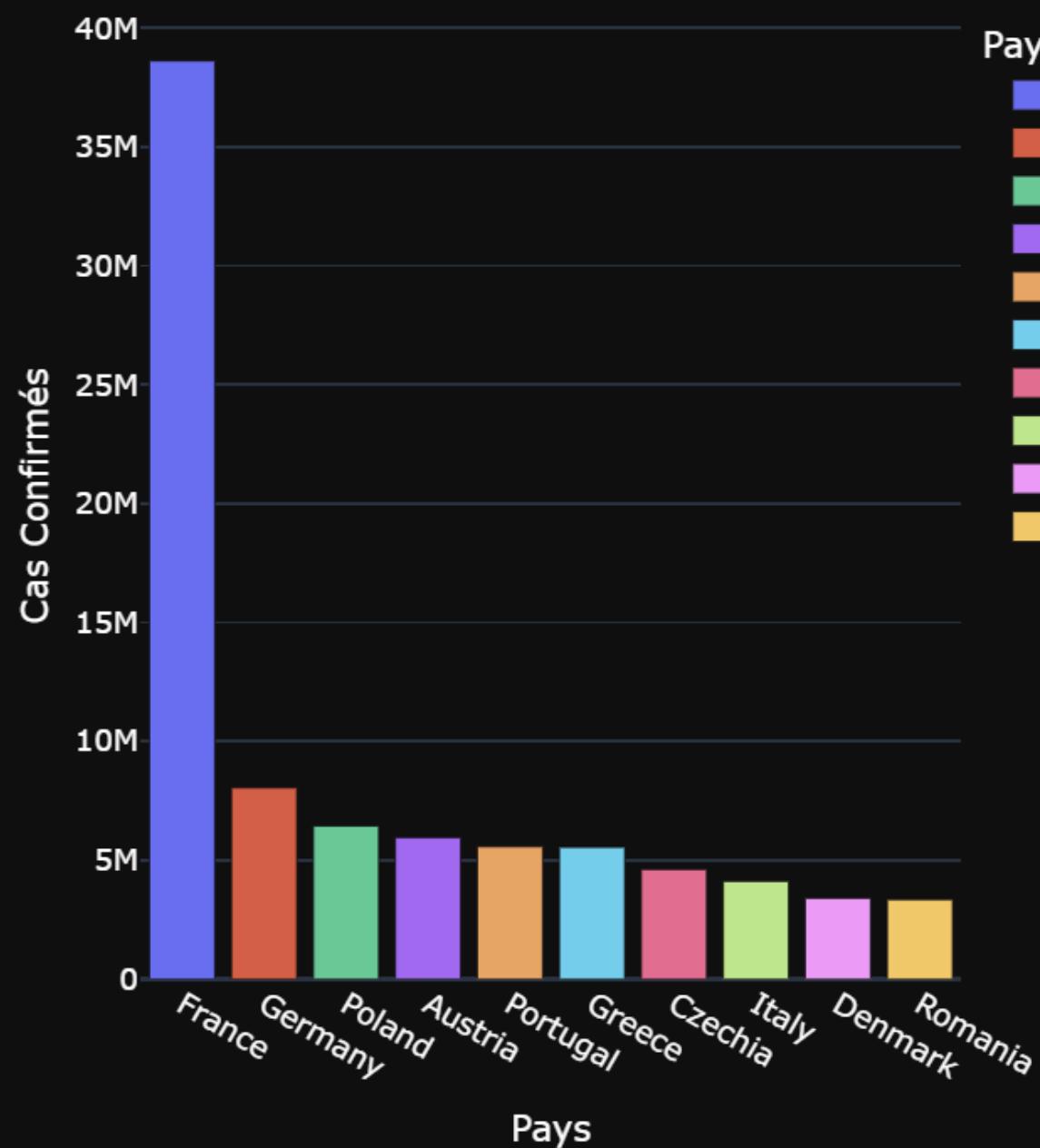
Tableau de bord COVID-19 - Top 10 Pays de l'UE

Sélectionnez une date :

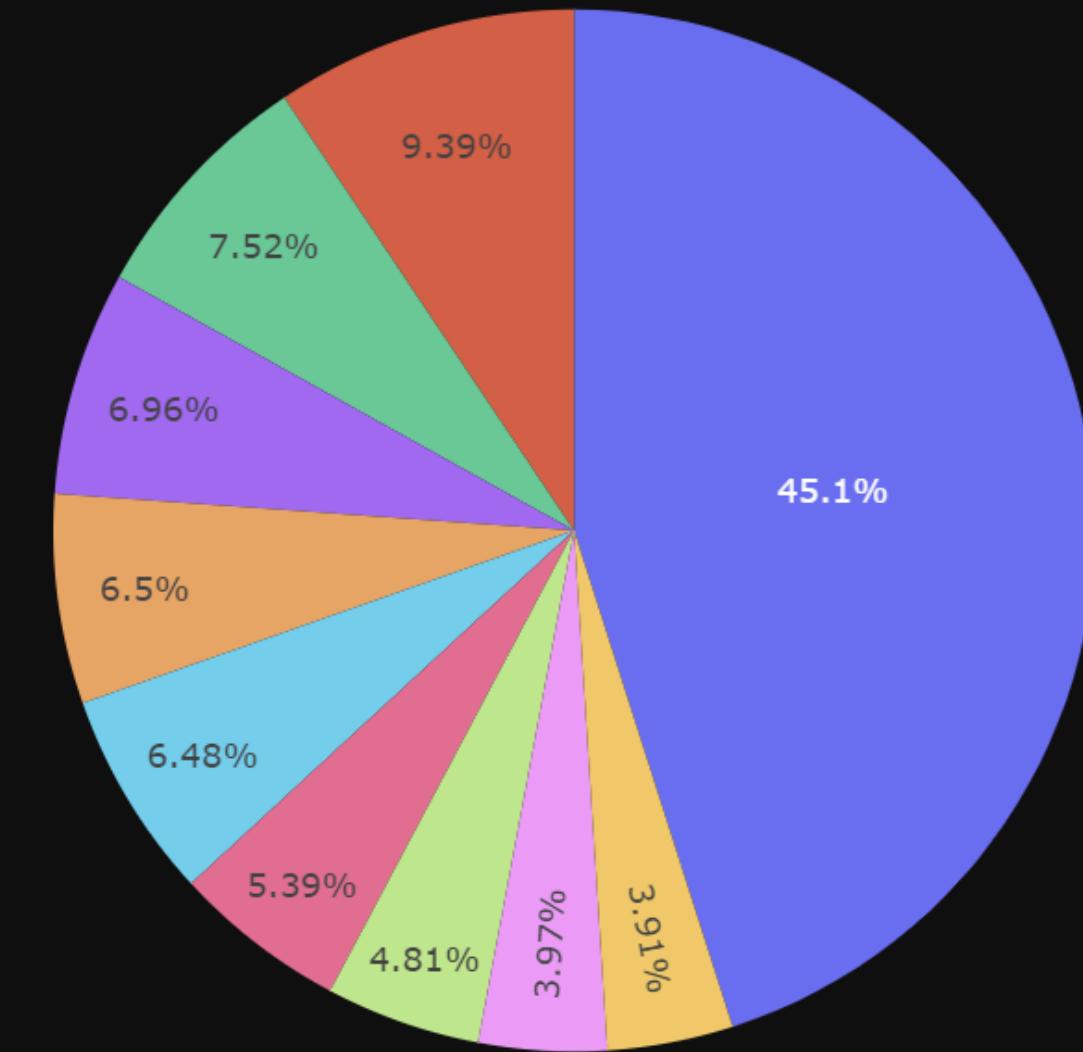


Top 10 des cas confirmés au 2023-03-10

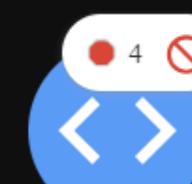
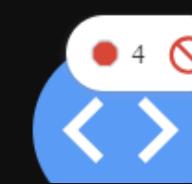


**Pays**

- France
- Germany
- Poland
- Austria
- Portugal
- Greece
- Czechia
- Italy
- Denmark
- Romania



- France
- Germany
- Poland
- Austria
- Portugal
- Greece
- Czechia
- Italy
- Denmark
- Romania



Rank	Country/Region	Confirmed
1	France	38618509
2	Germany	8048396
3	Poland	6444960
4	Austria	5961143
5	Portugal	5570473
6	Greece	5548487
7	Czechia	4618256
8	Italy	4118413
9	Denmark	3404407
10	Romania	3346046