Comparative Temporal Analysis of COVID-19 Data in Italy and France

February 14, 2025

1. Eleonora Basilico, 1090124, eleonora.basilico@edu.unito.it
2. Edoardo Carroccetto, 914565, edoardo.carroccetto@edu.unito.it
3. Paola Leone, 913652, paola.leone@edu.unito.it

1 Project description and libraries

This project aims to perform a comparative temporal analysis of COVID-19 data in Italy and France using Python. The analysis will leverage Pandas for data manipulation and Seaborn along with Matplotlib for visualization. The data will be retrieved from the COVID-19 Data API provided by disease.sh, focusing on understanding the temporal trends in key metrics such as cases, deaths, and recoveries.

We import below the necessary libraries.

[1]:
```
# Import necessary libraries
import requests
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2 Data retrieval and loading

We retrieve and load the data.

[2]:
```
resp = requests.get("https://disease.sh/v3/covid-19/historical/France?
 ↪lastdays=all")
resp.raise_for_status()
data = resp.json()

# Extract the 'timeline' section that contains the historical data
timeline = data['timeline']

# Turn timeline into a pandas DataFrame
df_france = pd.DataFrame(timeline)
df_france
```

[2]:

	cases	deaths	recovered
1/22/20	0	0	0
1/23/20	0	0	0
1/24/20	2	0	0
1/25/20	3	0	0
1/26/20	3	0	0
…	…	…	…
3/5/23	39839090	166071	0
3/6/23	39847236	166114	0
3/7/23	39854299	166138	0
3/8/23	39860410	166165	0
3/9/23	39866718	166176	0

[1143 rows x 3 columns]

[3]:
```
resp = requests.get("https://disease.sh/v3/covid-19/historical/Italy?
 ↪lastdays=all")
resp.raise_for_status()
data = resp.json()

# Extract the 'timeline' section which contains the historical data
timeline = data['timeline']

# Turn timeline into a pandas DataFrame
df_italy = pd.DataFrame(timeline)
df_italy
```

[3]:

	cases	deaths	recovered
1/22/20	0	0	0
1/23/20	0	0	0
1/24/20	0	0	0
1/25/20	0	0	0
1/26/20	0	0	0
…	…	…	…
3/5/23	25603510	188322	0
3/6/23	25603510	188322	0
3/7/23	25603510	188322	0
3/8/23	25603510	188322	0
3/9/23	25603510	188322	0

[1143 rows x 3 columns]

Both DataFrames contain three variables: `cases`, `deaths` and `recovered`. Since we don't have much information about how the data were collected, we assume they represent the cumulative numbers of COVID-19 cases, deaths, and recoveries in France and Italy respectively.

3 Data cleaning and preprocessing

First of all we check if there is any missing value.

[4]:
```
# Number of missing values in the two DataFrames
missing_values_france = df_france.isnull().sum()
missing_values_italy = df_italy.isnull().sum()

print('The number of missing values for France and Italy respectively is:')
print(missing_values_france,missing_values_italy, sep = "\n")
```

```
The number of missing values for France and Italy respectively is:
cases        0
deaths       0
recovered    0
dtype: int64
cases        0
deaths       0
recovered    0
dtype: int64
```

As we can see, there is no missing value.

Then, we verify the consistency of the column data types, i.e., we check that all columns contain only integer values.

[5]:
```
print('The column data types for France and Italy respectively are:')
print(df_france.dtypes, df_italy.dtypes, sep = "\n")
```

```
The column data types for France and Italy respectively are:
cases        int64
deaths       int64
recovered    int64
dtype: object
cases        int64
deaths       int64
recovered    int64
dtype: object
```

We convert indices into date format for clarity and proper time-based analysis, then we verify the change.

[6]:
```
# Convert indices into date format
df_france.index = pd.to_datetime(df_france.index)
df_italy.index = pd.to_datetime(df_italy.index)

print(df_france.index.dtype, df_italy.index.dtype, sep = "\n")
```

```
datetime64[ns]
datetime64[ns]
```

We now check if there are rows with the same index.

[7]:
```
print('All indices in df_france are unique:', df_france.index.is_unique)
print('All indices in df_italy are unique:', df_italy.index.is_unique)
```

```
All indices in df_france are unique: True
All indices in df_italy are unique: True
```

As we can see, all indices are unique.

We also need to ensure the non-negativity of all variables.

[8]:
```
# Check if the variables 'cases', 'deaths', and 'recovered' are non-negative
if (df_france < 0).values.any():
    print("There are negative values!")
else:
    print("There are no negative values in df_france.")

if (df_italy < 0).values.any():
    print("There are negative values!")
else:
    print("There are no negative values in df_italy.")
```

```
There are no negative values in df_france.
There are no negative values in df_italy.
```

Now, since the variables are cumulative, we need to verify that in both DataFrames:

- For all variables each value is greater or equal than the value from the previous day.
- The number of deaths and recoveries is always lower than the number of cases.

To verify that for all variables each value is greater than the value from the previous day, we create two new DataFrames (`df_france_daily` and `df_italy_daily`), which contain the difference between each value and the previous one. Since the variables are cumulative, these differences represent daily cases, deaths, and recoveries, respectively. We need to ensure that in these two new DataFrames there are no negative values.

[9]:
```
# Construction of DataFrames with daily cases, deaths and recoveries
df_france_daily = df_france.diff().dropna()
df_italy_daily = df_italy.diff().dropna()
```

[10]:
```
# Count of negative values in df_france_daily and df_italy_daily
n_neg_val_france = (df_france_daily < 0).sum()
n_neg_val_italy = (df_italy_daily < 0).sum()

print('The number of negative values in df_france_daily and df_italy_daily is␣
 ↪respectively:')
print(n_neg_val_france, n_neg_val_italy, sep = "\n")
```

```
The number of negative values in df_france_daily and df_italy_daily is
respectively:
```

```
cases        12
deaths        8
recovered     4
dtype: int64
cases        1
deaths       1
recovered    3
dtype: int64
```

As we can see, in both DataFrames there are negative values. Let us analyse Italy and France separately to understand how to address the issues. We start with Italy, as it has fewer negative values.

To better analyse the problem, we print a 9-days window of `df_italy` around each date where a negative value appears in the DataFrame `df_italy_daily`.

[11]:
```
# Number of rows to display before and after a negative value
# pd.DateOffset is used because we are working with date indices
window = pd.DateOffset(days = 4)

for col in ['cases', 'deaths', 'recovered']:
    # Find rows of df_italy_daily with negative values
    neg_rows = df_italy_daily[df_italy_daily[col] < 0]
    for date in neg_rows.index:
        # Print the section of df_italy where the cumulative trend is violated
        print(f"\nNegative value in '{col}' on {date}:")
        print(df_italy.loc[date - window : date + window])
```

```
Negative value in 'cases' on 2020-06-19 00:00:00:
             cases  deaths  recovered
2020-06-15  237290   34371     177010
2020-06-16  237500   34405     178526
2020-06-17  237828   34448     179455
2020-06-18  238159   34514     180544
2020-06-19  238011   34561     181907
2020-06-20  238275   34610     182453
2020-06-21  238499   34634     182893
2020-06-22  238720   34657     183426
2020-06-23  238833   34675     184585

Negative value in 'deaths' on 2020-06-24 00:00:00:
             cases  deaths  recovered
2020-06-20  238275   34610     182453
2020-06-21  238499   34634     182893
2020-06-22  238720   34657     183426
2020-06-23  238833   34675     184585
2020-06-24  239410   34644     186111
2020-06-25  239706   34678     186725
```

```
2020-06-26  239961   34708     187615
2020-06-27  240136   34716     188584
2020-06-28  240310   34738     188891

Negative value in 'recovered' on 2020-02-24 00:00:00:
            cases  deaths  recovered
2020-02-20      3       0          0
2020-02-21     20       1          0
2020-02-22     62       2          1
2020-02-23    155       3          2
2020-02-24    229       7          1
2020-02-25    322      10          1
2020-02-26    453      12          3
2020-02-27    655      17         45
2020-02-28    888      21         46

Negative value in 'recovered' on 2020-08-31 00:00:00:
             cases  deaths  recovered
2020-08-27  263949   35463     206554
2020-08-28  265409   35472     206902
2020-08-29  266853   35473     208224
2020-08-30  268218   35477     208536
2020-08-31  269214   35483     207653
2020-09-01  270189   35491     207944
2020-09-02  271515   35497     208201
2020-09-03  272912   35507     208490
2020-09-04  274644   35518     209027

Negative value in 'recovered' on 2021-08-05 00:00:00:
              cases  deaths  recovered
2021-08-01  4355348  128068    4135930
2021-08-02  4358533  128088    4137428
2021-08-03  4363374  128115    4141043
2021-08-04  4369964  128136    4144608
2021-08-05  4377188  128163          0
2021-08-06  4383787  128187          0
2021-08-07  4390684  128209          0
2021-08-08  4396417  128220          0
2021-08-09  4400617  128242          0
```

We can notice that:

- In most cases, the anomaly occurs for a single day, but even when the error propagates over multiple days, as in the second-to-last case, the values remain relatively close to each other. Therefore, it makes sense to replace each anomaly with the previous valid value and maintain it until a larger value appears. This approach is simple but effective: it preserves the overall trend while keeping the cumulative nature of the data intact, and prevents small inconsistencies from affecting the analysis.

- It seems that after 04-08-2021 the variable `recovered` is always 0.

Before addressing the issue of negative values, let us first check whether all values for the variable `recovered` are indeed 0 after 04-08-2021.

[12]:
```
if (df_italy.loc['2021-08-05':, 'recovered'] != 0).any():
    print("There are non-zero values!")
else:
    print("All values after 04-08-2021 are 0 in df_italy.")
```

```
All values after 04-08-2021 are 0 in df_italy.
```

Since all values for the variable `recovered` are 0 after 04-08-2021, we exclude this variable from our analysis beyond this date. Therefore, we create two separate DataFrames: one containing all variables up to 04-08-2021 and another containing only `cases` and `deaths` for the entire time period of the original DataFrame `df_italy`. From now on, we will analyse them separately.

[13]:
```
# We define the DataFrame containing all variables up to 04-08-2021
df_italy_1 = df_italy.loc[:'2021-08-04']
df_italy_1
```

[13]:

	cases	deaths	recovered
2020-01-22	0	0	0
2020-01-23	0	0	0
2020-01-24	0	0	0
2020-01-25	0	0	0
2020-01-26	0	0	0
…	…	…	…
2021-07-31	4350028	128063	4134680
2021-08-01	4355348	128068	4135930
2021-08-02	4358533	128088	4137428
2021-08-03	4363374	128115	4141043
2021-08-04	4369964	128136	4144608

[561 rows x 3 columns]

[14]:
```
# We define the DataFrame containing only cases and deaths for the entire time␣
↪period
df_italy_2 = df_italy.drop('recovered', axis = 1)
df_italy_2
```

[14]:

	cases	deaths
2020-01-22	0	0
2020-01-23	0	0
2020-01-24	0	0
2020-01-25	0	0
2020-01-26	0	0
…	…	…
2023-03-05	25603510	188322

```
2023-03-06  25603510  188322
2023-03-07  25603510  188322
2023-03-08  25603510  188322
2023-03-09  25603510  188322

[1143 rows x 2 columns]
```

We can now proceed with solving the issue of negative values. To do so, we apply to both DataFrames the `.cummax()` function, which ensures that each value is at least as large as the previous one.

[15]:
```
df_italy_1 = df_italy_1.cummax()
df_italy_2 = df_italy_2.cummax()
```

Let us check that the corresponding DataFrames containing the daily cases, deaths and recoveries don't have any negative value now.

[16]:
```
# Construction of DataFrames with daily cases, deaths and recoveries
df_italy_1_daily = df_italy_1.diff().dropna()
df_italy_2_daily = df_italy_2.diff().dropna()

# Count of negative values in df_italy_1_daily and df_italy_2_daily
n_neg_val_italy_1 = (df_italy_1_daily < 0).sum()
n_neg_val_italy_2 = (df_italy_2_daily < 0).sum()

print('The number of negative values in df_italy_1_daily and df_italy_2_daily␣
 ↪is respectively:')
print(n_neg_val_italy_1, n_neg_val_italy_2, sep = "\n")
```

```
The number of negative values in df_italy_1_daily and df_italy_2_daily is
respectively:
cases        0
deaths       0
recovered    0
dtype: int64
cases     0
deaths    0
dtype: int64
```

We now use the same approach for the `df_france` DataFrame. So, we start by printing a 9-days window of `df_france` around each date where a negative value appears in the DataFrame `df_france_daily`.

[17]:
```
# Number of rows to display before and after a negative value
# pd.DateOffset is used because we are working with date indices
window = pd.DateOffset(days = 4)

for col in ['cases', 'deaths', 'recovered']:
    # Find rows of df_france_daily with negative values
```

```
    neg_rows = df_france_daily[df_france_daily[col] < 0]
    for date in neg_rows.index:
        # Print the section of df_france where the cumulative trend is violated
        print(f"\nNegative value in '{col}' on {date}:")
        print(df_france.loc[date - window : date + window])
```

```
Negative value in 'cases' on 2020-04-04 00:00:00:
            cases  deaths  recovered
2020-03-31  52281    3526       9513
2020-04-01  57125    4779      11053
2020-04-02  59227    5388      12548
2020-04-03  64452    6510      14135
2020-04-04  47376    7562      15572
2020-04-05  48225    8081      16349
2020-04-06  50884    8914      17428
2020-04-07  47395   10330      19523
2020-04-08  51250   10874      21452

Negative value in 'cases' on 2020-04-07 00:00:00:
            cases  deaths  recovered
2020-04-03  64452    6510      14135
2020-04-04  47376    7562      15572
2020-04-05  48225    8081      16349
2020-04-06  50884    8914      17428
2020-04-07  47395   10330      19523
2020-04-08  51250   10874      21452
2020-04-09  55034   12214      23413
2020-04-10  56600   13199      25195
2020-04-11  58045   13835      26663

Negative value in 'cases' on 2020-04-23 00:00:00:
             cases  deaths  recovered
2020-04-19  151955   19694      36578
2020-04-20  154402   20241      37409
2020-04-21  157068   20769      39181
2020-04-22  158867   21313      40657
2020-04-23  157158   21829      42088
2020-04-24  159969   22218      43493
2020-04-25  161647   22587      44594
2020-04-26  162280   22830      44903
2020-04-27  165966   23267      45513

Negative value in 'cases' on 2020-04-29 00:00:00:
             cases  deaths  recovered
2020-04-25  161647   22587      44594
2020-04-26  162280   22830      44903
2020-04-27  165966   23267      45513
```

```
2020-04-28  169098   23634      46886
2020-04-29  167643   24060      48228
2020-04-30  168861   24349      49476
2020-05-01  169387   24566      50212
2020-05-02  170179   24763      50663
2020-05-03  170540   24898      50885

Negative value in 'cases' on 2020-05-24 00:00:00:
             cases  deaths  recovered
2020-05-20  183129   28136      63472
2020-05-21  183396   28218      63976
2020-05-22  184152   28366      64327
2020-05-23  184698   28452      64665
2020-05-24  184259   28372      64735
2020-05-25  184584   28461      65317
2020-05-26  184839   28598      65997
2020-05-27  185012   28599      66702
2020-05-28  188355   28666      67309

Negative value in 'cases' on 2020-06-02 00:00:00:
             cases  deaths  recovered
2020-05-29  188949   28717      67921
2020-05-30  190743   28774      68386
2020-05-31  190975   28805      68473
2020-06-01  191382   28837      68558
2020-06-02  190735   28943      68930
2020-06-03  187509   29024      69573
2020-06-04  191869   29069      70094
2020-06-05  192450   29114      70622
2020-06-06  193022   29145      70924

Negative value in 'cases' on 2020-06-03 00:00:00:
             cases  deaths  recovered
2020-05-30  190743   28774      68386
2020-05-31  190975   28805      68473
2020-06-01  191382   28837      68558
2020-06-02  190735   28943      68930
2020-06-03  187509   29024      69573
2020-06-04  191869   29069      70094
2020-06-05  192450   29114      70622
2020-06-06  193022   29145      70924
2020-06-07  193363   29158      70961

Negative value in 'cases' on 2020-06-28 00:00:00:
             cases  deaths  recovered
2020-06-24  201598   29735      75251
2020-06-25  201853   29754      75475
2020-06-26  203116   29780      75773
```

```
2020-06-27  203564   29781      75773
2020-06-28  203157   29781      75774
2020-06-29  203802   29816      76124
2020-06-30  204244   29846      76399
2020-07-01  205234   29865      76674
2020-07-02  205773   29878      76927

Negative value in 'cases' on 2020-11-04 00:00:00:
              cases  deaths  recovered
2020-10-31  1414396   36827     123095
2020-11-01  1460575   37058     123664
2020-11-02  1566666   37486     123664
2020-11-03  1639267   38765     123664
2020-11-04  1593146   38731     126195
2020-11-05  1650966   39088     129759
2020-11-06  1711918   39917     131810
2020-11-07  1798573   40220     133419
2020-11-08  1837135   40490     134095

Negative value in 'cases' on 2021-05-20 00:00:00:
              cases  deaths  recovered
2021-05-16  5939130  107779     378958
2021-05-17  5942481  107975     379812
2021-05-18  5959704  108203     381251
2021-05-19  5978761  108344     382501
2021-05-20  5629921  108477     382501
2021-05-21  5642724  108600     382517
2021-05-22  5655335  108689     382517
2021-05-23  5665101  108760     382519
2021-05-24  5667331  108822     385961

Negative value in 'cases' on 2022-05-23 00:00:00:
               cases  deaths  recovered
2022-05-19  29517146  148753          0
2022-05-20  29541498  148818          0
2022-05-21  29564005  148820          0
2022-05-22  29580970  148823          0
2022-05-23  29551335  148955          0
2022-05-24  29583616  149044          0
2022-05-25  29605758  149095          0
2022-05-26  29626992  149123          0
2022-05-27  29631816  149168          0

Negative value in 'cases' on 2022-06-11 00:00:00:
               cases  deaths  recovered
2022-06-07  29852463  149630          0
2022-06-08  29906452  149707          0
2022-06-09  29946603  149749          0
```

```
2022-06-10  29946697  149749          0
2022-06-11  29946603  149788          0
2022-06-12  29946603  149788          0
2022-06-13  29946647  149788          0
2022-06-14  30123426  149937          0
2022-06-15  30175534  149986          0

Negative value in 'deaths' on 2020-05-19 00:00:00:
             cases  deaths  recovered
2020-05-15  181148   27532      60562
2020-05-16  181563   27630      60562
2020-05-17  181703   28113      61327
2020-05-18  182147   28242      61843
2020-05-19  182648   28025      62678
2020-05-20  183129   28136      63472
2020-05-21  183396   28218      63976
2020-05-22  184152   28366      64327
2020-05-23  184698   28452      64665

Negative value in 'deaths' on 2020-05-24 00:00:00:
             cases  deaths  recovered
2020-05-20  183129   28136      63472
2020-05-21  183396   28218      63976
2020-05-22  184152   28366      64327
2020-05-23  184698   28452      64665
2020-05-24  184259   28372      64735
2020-05-25  184584   28461      65317
2020-05-26  184839   28598      65997
2020-05-27  185012   28599      66702
2020-05-28  188355   28666      67309

Negative value in 'deaths' on 2020-07-05 00:00:00:
             cases  deaths  recovered
2020-07-01  205234   29865      76674
2020-07-02  205773   29878      76927
2020-07-03  206312   29896      77185
2020-07-04  206670   29896      77185
2020-07-05  206682   29895      77185
2020-07-06  207700   29925      77444
2020-07-07  208154   29936      77780
2020-07-08  208976   29967      77780
2020-07-09  209420   29983      78295

Negative value in 'deaths' on 2020-07-21 00:00:00:
             cases  deaths  recovered
2020-07-17  214009   30155      79371
2020-07-18  214174   30158      79371
2020-07-19  214178   30158      79371
```

```
2020-07-20  216089   30182      79668
2020-07-21  216684   30169      79861
2020-07-22  217605   30175      80084
2020-07-23  218841   30186      80600
2020-07-24  219932   30196      80943
2020-07-25  220016   30196      80945

Negative value in 'deaths' on 2020-09-04 00:00:00:
             cases  deaths  recovered
2020-08-31  321160   30646      86790
2020-09-01  326264   30673      87036
2020-09-02  333351   30699      87418
2020-09-03  340473   30716      87661
2020-09-04  349333   30696      87927
2020-09-05  357927   30708      87927
2020-09-06  364943   30712      87927
2020-09-07  369209   30735      88484
2020-09-08  375947   30773      88876

Negative value in 'deaths' on 2020-10-25 00:00:00:
              cases  deaths  recovered
2020-10-21  1002624   34080     111715
2020-10-22  1043437   34234     112662
2020-10-23  1086286   34534     113636
2020-10-24  1131551   34670     113636
2020-10-25  1138167   34649     113698
2020-10-26  1211177   35038     115964
2020-10-27  1245635   35566     117400
2020-10-28  1283185   35825     119413
2020-10-29  1329821   36058     120723

Negative value in 'deaths' on 2020-11-04 00:00:00:
              cases  deaths  recovered
2020-10-31  1414396   36827     123095
2020-11-01  1460575   37058     123664
2020-11-02  1566666   37486     123664
2020-11-03  1639267   38765     123664
2020-11-04  1593146   38731     126195
2020-11-05  1650966   39088     129759
2020-11-06  1711918   39917     131810
2020-11-07  1798573   40220     133419
2020-11-08  1837135   40490     134095

Negative value in 'deaths' on 2023-01-25 00:00:00:
               cases  deaths  recovered
2023-01-21  39681509  164793          0
2023-01-22  39681509  164793          0
2023-01-23  39690337  164990          0
```

```
2023-01-24  39697613  165049          0
2023-01-25  39702995  165027          0
2023-01-26  39708282  165077          0
2023-01-27  39712963  165121          0
2023-01-28  39713004  165121          0
2023-01-29  39713004  165121          0

Negative value in 'recovered' on 2020-09-14 00:00:00:
             cases  deaths  recovered
2020-09-10  394308   30824      89468
2020-09-11  403837   30906      89890
2020-09-12  415174   30924      90445
2020-09-13  421519   30929      90445
2020-09-14  427839   30963      90432
2020-09-15  435665   31013      90816
2020-09-16  445932   31061      91293
2020-09-17  456171   31108      91765
2020-09-18  469404   31262      92700

Negative value in 'recovered' on 2020-09-24 00:00:00:
             cases  deaths  recovered
2020-09-20  493500   31298      93586
2020-09-21  499039   31351      94289
2020-09-22  509123   31430      94961
2020-09-23  522355   31476      96498
2020-09-24  538264   31524      95980
2020-09-25  554368   31675      96909
2020-09-26  568588   31714      96937
2020-09-27  579223   31741      96937
2020-09-28  583522   31825      97127

Negative value in 'recovered' on 2021-07-28 00:00:00:
              cases  deaths  recovered
2021-07-24  6041313  111800     410525
2021-07-25  6056555  111806     410554
2021-07-26  6061969  111852     410903
2021-07-27  6089097  111881     413226
2021-07-28  6117020  111925     411797
2021-07-29  6142449  111954     412373
2021-07-30  6166759  112017     412805
2021-07-31  6190621  112061     413170
2021-08-01  6210335  112079     413278

Negative value in 'recovered' on 2021-08-05 00:00:00:
              cases  deaths  recovered
2021-08-01  6210335  112079     413278
2021-08-02  6218927  112136     413357
2021-08-03  6243225  112196     414433
```

```
2021-08-04  6272466  112245     415111
2021-08-05  6299415  112298          0
2021-08-06  6325146  112364          0
2021-08-07  6351003  112396          0
2021-08-08  6371463  112427          0
2021-08-09  6378681  112511          0
```

As with `df_italy`, also with `df_france` most anomalies occur for a single day, and when the error propagates over multiple days, the values mostly remain relatively close to each other (there are only few exceptions). Therefore, once again, we replace each anomaly with the previous valid value and maintain it until a larger value appears. Furthermore, it appears that after 04-08-2021 the variable recovered is always 0.

Let us verify whether all values for the variable `recovered` are indeed 0 after 04-08-2021.

[18]:
```
if (df_france.loc['2021-08-05':, 'recovered'] != 0).any():
    print("There are non-zero values!")
else:
    print("All values after 04-08-2021 are 0 in df_france.")
```

```
All values after 04-08-2021 are 0 in df_france.
```

Since all values for the variable `recovered` are 0 after 04-08-2021, we must exclude this variable from our analysis beyond this date. As before, we create two separate DataFrames: one containing all variables up to 04-08-2021 and another containing only `cases` and `deaths` for the entire time period of the original DataFrame `df_france`.

[19]:
```
# We define the DataFrame containing all variables up to 04-08-2021
df_france_1 = df_france.loc[:'2021-08-04']
df_france_1
```

[19]:
```
              cases  deaths  recovered
2020-01-22        0       0          0
2020-01-23        0       0          0
2020-01-24        2       0          0
2020-01-25        3       0          0
2020-01-26        3       0          0
…               …     …          …
2021-07-31  6190621  112061     413170
2021-08-01  6210335  112079     413278
2021-08-02  6218927  112136     413357
2021-08-03  6243225  112196     414433
2021-08-04  6272466  112245     415111

[561 rows x 3 columns]
```

[20]:
```
# We define the DataFrame containing only cases and deaths for the entire time␣
 ↪period
df_france_2 = df_france.drop('recovered', axis = 1)
df_france_2
```

[20]:
```
                cases  deaths
2020-01-22          0       0
2020-01-23          0       0
2020-01-24          2       0
2020-01-25          3       0
2020-01-26          3       0
…                   …       …
2023-03-05   39839090  166071
2023-03-06   39847236  166114
2023-03-07   39854299  166138
2023-03-08   39860410  166165
2023-03-09   39866718  166176

[1143 rows x 2 columns]
```

We now solve the problem of negative values. As before, we apply to both DataFrames the `.cummax()` function. Then, we verify that the corresponding DataFrames containing the daily cases, deaths and recoveries don't have negative values anymore.

[21]:
```
df_france_1 = df_france_1.cummax()
df_france_2 = df_france_2.cummax()
```

[22]:
```
# Construction of DataFrames with daily cases, deaths and recoveries
df_france_1_daily = df_france_1.diff().dropna()
df_france_2_daily = df_france_2.diff().dropna()

# Count of negative values in df_france_1_daily and df_france_2_daily
n_neg_val_france_1 = (df_france_1_daily < 0).sum()
n_neg_val_france_2 = (df_france_2_daily < 0).sum()

print('The number of negative values in df_france_1_daily and df_france_2_daily␣
 ↪is respectively:')
print(n_neg_val_france_1, n_neg_val_france_2, sep = "\n")
```

```
The number of negative values in df_france_1_daily and df_france_2_daily is
respectively:
cases        0
deaths       0
recovered    0
dtype: int64
cases     0
deaths    0
dtype: int64
```

Finally, we need to verify if the number of deaths and recoveries is always lower than the number of cases. We do it for all the DataFrames. We can notice that it is enough to verify the column `deaths` only for the DataFrames `df_france_2` and `df_italy_2`, as they contain the values of the whole time period.

[23]:
```
if not (df_france_2['deaths'] <= df_france_2['cases']).all():
    print("There are inconsistencies in the number of deaths!")
else:
    print("The number of deaths is always lower than the number of cases in␣
 ↪df_france_2.")

if not (df_italy_2['deaths'] <= df_italy_2['cases']).all():
    print("There are inconsistencies in the number of deaths!")
else:
    print("The number of deaths is always lower than the number of cases in␣
 ↪df_italy_2.")
```

```
The number of deaths is always lower than the number of cases in df_france_2.
The number of deaths is always lower than the number of cases in df_italy_2.
```

[24]:
```
if not (df_france_1['recovered'] <= df_france_1['cases']).all():
    print("There are inconsistencies in the number of recoveries!")
else:
    print("The number of recoveries is always lower than the number of cases in␣
 ↪df_france_1.")

if not (df_italy_1['recovered'] <= df_italy_1['cases']).all():
    print("There are inconsistencies in the number of recoveries!")
else:
    print("The number of recoveries is always lower than the number of cases in␣
 ↪df_italy_1.")
```

```
The number of recoveries is always lower than the number of cases in
df_france_1.
The number of recoveries is always lower than the number of cases in df_italy_1.
```

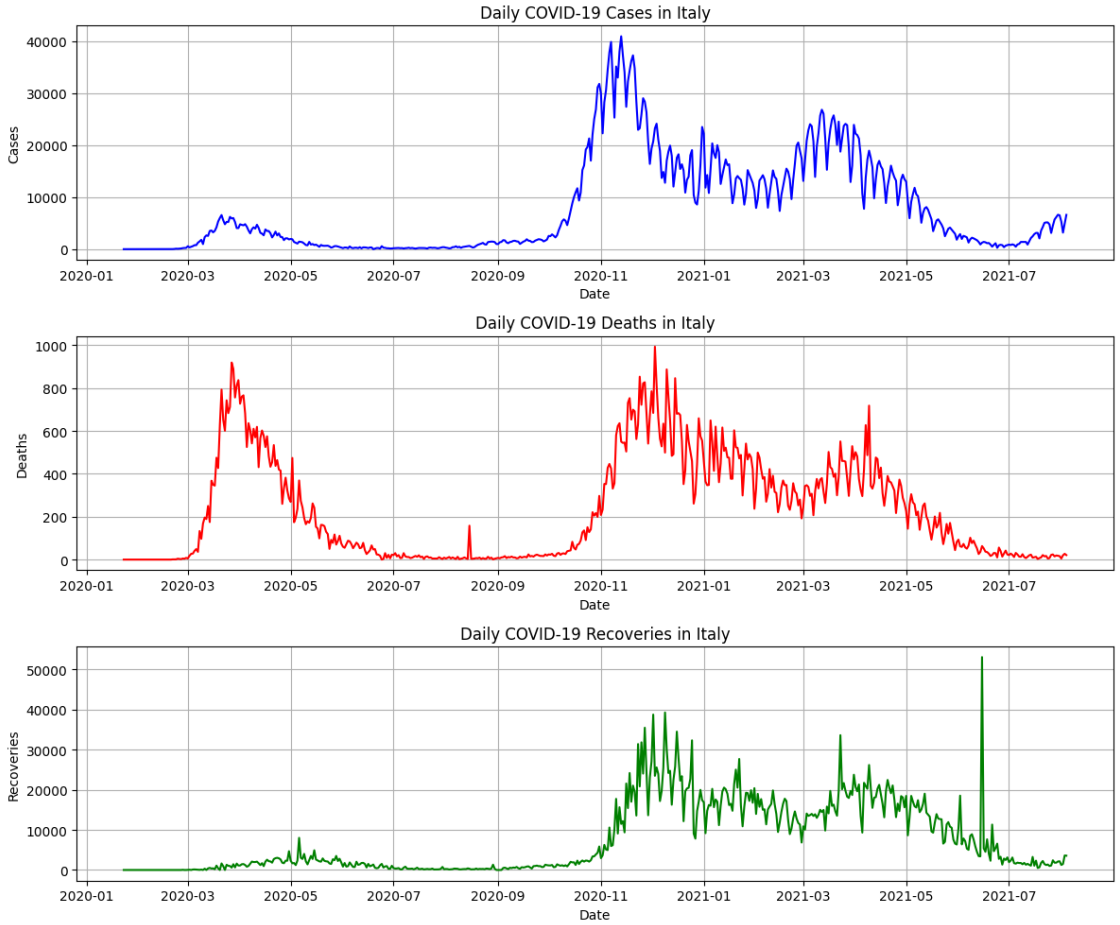
4 Temporal trends analysis

In this section, we analyse the temporal evolution of COVID-19 cases, deaths and recoveries in Italy and France separately. We will compute and visualize both daily and weekly trends, and we will do it for both DataFrames, the one containing all variables up to 04-08-2021 and the one including only `cases` and `deaths` for the whole time period. The goal is to provide a comprehensive view of the pandemic's progression in Italy and France.

4.1 Italy

We begin by analyzing the DataFrame that includes all variables up to 04-08-2021. First, we examine the daily trends of the three variables. This information is stored in the DataFrame `df_italy_1_daily`, which contains the daily cases, deaths, and recoveries up to 04-08-2021. To explore short-term fluctuations and better understand the virus's daily dynamics, we plot each variable separately.

[25]:
```
# Create a figure with three subplots
fig, axes = plt.subplots(3, 1, figsize = (12, 10))

# Plot Daily Cases
sns.lineplot(ax = axes[0], data = df_italy_1_daily, x = df_italy_1_daily.index,␣
 ↪y = df_italy_1_daily["cases"], color = 'blue')
axes[0].set_title("Daily COVID-19 Cases in Italy")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Daily Deaths
sns.lineplot(ax = axes[1], data = df_italy_1_daily, x = df_italy_1_daily.index,␣
 ↪y = df_italy_1_daily["deaths"], color = 'red')
axes[1].set_title("Daily COVID-19 Deaths in Italy")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

# Plot Daily Recoveries
sns.lineplot(ax = axes[2], data = df_italy_1_daily, x = df_italy_1_daily.index,␣
 ↪y = df_italy_1_daily["recovered"], color = 'green')
axes[2].set_title("Daily COVID-19 Recoveries in Italy")
axes[2].set_xlabel("Date")
axes[2].set_ylabel("Recoveries")
axes[2].grid(True)

plt.tight_layout()
plt.show()
```



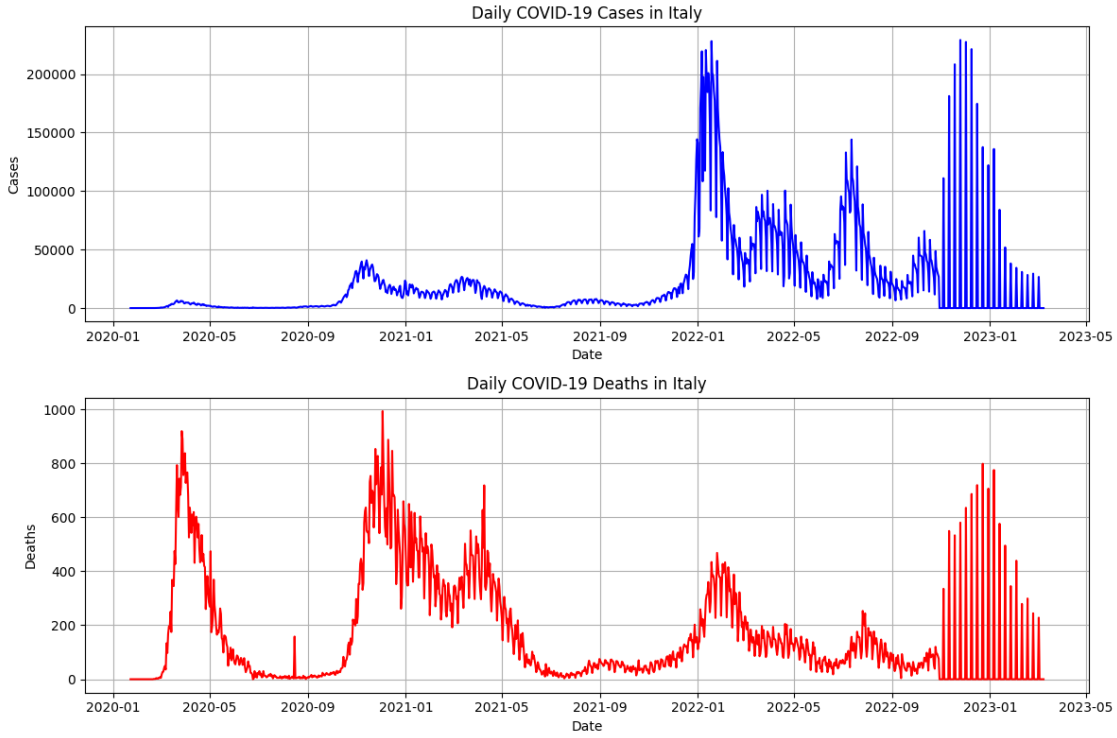
The line plots above show the daily trends for COVID-19 metrics in Italy from 22-01-2020 to 04-08-2021. This period covers the first three pandemic waves, characterized by increased infections, hospitalizations, and deaths. Specifically:

- *First Wave* (March 2020 – May 2020): The daily case count peaked at approximately $7,000$ cases, with a maximum of around 900 daily deaths and $8,000$ daily recoveries. While the impact of the virus was severe, the total number of reported infections remained relatively low compared to later waves. This was likely due to more limited testing capabilities and the strict lockdown measures implemented early on.
- *Second Wave* (October 2020 – January 2021): This wave was significantly larger in scale. Daily cases surged to nearly eight times those of the first wave, reaching peaks of over $40,000$ cases per day. Despite the dramatic increase in infections, the number of daily deaths remained comparable to the first wave, suggesting improvements in medical treatments and increased hospital capacity. Recoveries also increased substantially, peaking at around $40,000$ per day, reflecting both the higher number of infections and possibly more efficient treatment protocols.
- *Third Wave* (March 2021 – May 2021): The peaks observed in this wave were slightly lower than those of the second wave, likely due to the effects of vaccination campaigns and the emergence of less severe variants.

Notably, the ratio of deaths to cases was significantly higher during the first wave compared to the subsequent waves. This can be attributed to factors such as improved testing, healthcare adaptation, the implementation of restrictions, and changes in public behavior.

Let us now analyse the DataFrame containing only the variables `cases` and `deaths` for the whole time period (`df_italy_2_daily`). This will allow us to gain a more comprehensive understanding of the long-term progression of these two metrics and will help us see how the situation evolved beyond July 2021.

[ ]:
```
# Create a figure with two subplots
fig, axes = plt.subplots(2, 1, figsize = (12, 8))

# Plot Daily Cases
sns.lineplot(ax = axes[0], data = df_italy_2_daily, x = df_italy_2_daily.index,␣
 ↪y = df_italy_2_daily["cases"], color = 'blue')
axes[0].set_title("Daily COVID-19 Cases in Italy")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Daily Deaths
sns.lineplot(ax = axes[1], data = df_italy_2_daily, x = df_italy_2_daily.index,␣
 ↪y = df_italy_2_daily["deaths"], color = 'red')
axes[1].set_title("Daily COVID-19 Deaths in Italy")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



The line plots above cover the entire period from January 2020 to March 2023. We can observe that:

- Additional waves occurred after 04-08-2021, with daily cases rising significantly compared to the first three waves. A peak of approximately 220, 000 daily cases was recorded after January 2022, far exceeding previous numbers. However, despite this sharp increase in infections, the number of daily deaths remained lower than in the previous waves. This trend suggests several possible factors at play, including widespread vaccination, improved medical treatments, and possible changes in the virus's characteristics, making later variants more transmissible but less lethal.
- The number of cases tends to decline during the summer months. This could be attributed to several factors, including increased outdoor activities, school and workplace closures, and higher temperatures, which may influence virus transmission.

Additionally, a strange pattern appears in the later period, where both variables repeatedly drop to zero. To investigate this anomaly, we print the last three weeks of data.

[27]:
```
df_italy_2_daily.tail(21)
```

[27]:

	cases	deaths
2023-02-17	28347.0	299.0
2023-02-18	0.0	0.0
2023-02-19	0.0	0.0
2023-02-20	0.0	0.0

```
2023-02-21      0.0     0.0
2023-02-22      0.0     0.0
2023-02-23      0.0     0.0
2023-02-24  29438.0   244.0
2023-02-25      0.0     0.0
2023-02-26      0.0     0.0
2023-02-27      0.0     0.0
2023-02-28      0.0     0.0
2023-03-01      0.0     0.0
2023-03-02      0.0     0.0
2023-03-03  26658.0   228.0
2023-03-04      0.0     0.0
2023-03-05      0.0     0.0
2023-03-06      0.0     0.0
2023-03-07      0.0     0.0
2023-03-08      0.0     0.0
2023-03-09      0.0     0.0
```

As we can see, this irregular pattern is due to the data being updated only once a week. It can be observed that this change occurs approximately from the end of October 2022.

We now aggregate the data on a weekly basis to smooth out the daily fluctuations and highlight long-term trends. For this purpose, we create two new DataFrames:

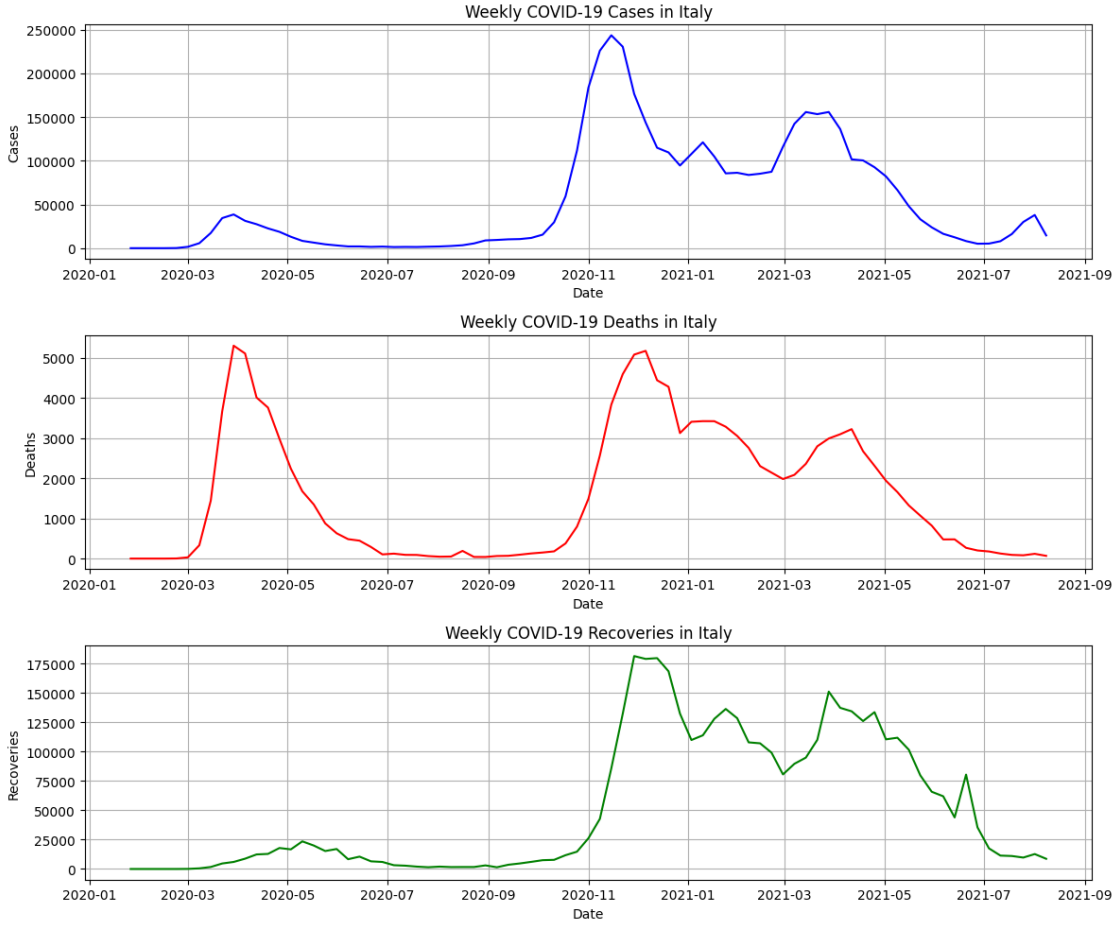
- `df_italy_weekly_1`: Aggregating data from `df_italy_1_daily`, which includes cases, deaths, and recoveries up to August 2021.
- `df_italy_weekly_2`: Aggregating data from `df_italy_2_daily`, which contains cases and deaths up to March 2023.

[28]:
```
# Perform weekly aggregation by summing the daily values within each week
df_italy_weekly_1 = df_italy_1_daily.resample('W').sum()
df_italy_weekly_2 = df_italy_2_daily.resample('W').sum()
```

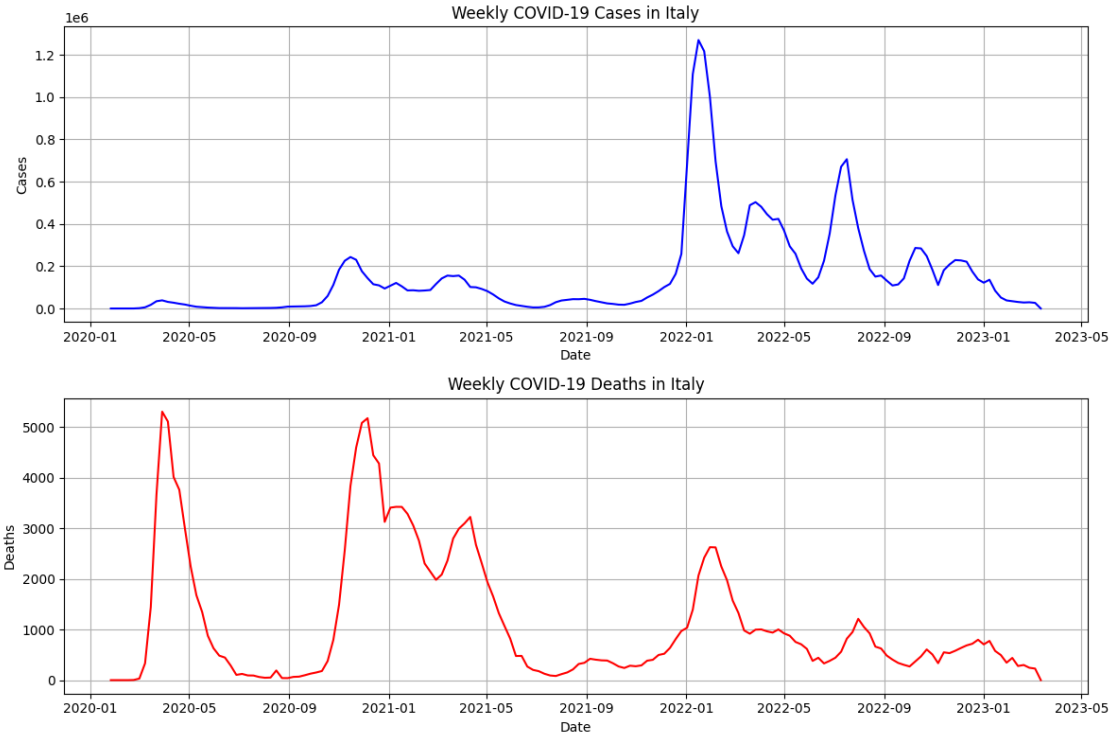
We plot the weekly trends for both datasets below.

[29]:
```
# Create a figure with three subplots
fig, axes = plt.subplots(3, 1, figsize = (12, 10))

# Plot Weekly Cases
sns.lineplot(ax = axes[0], data = df_italy_weekly_1, x = df_italy_weekly_1.
 ↪index, y = df_italy_weekly_1["cases"], color = 'blue')
axes[0].set_title("Weekly COVID-19 Cases in Italy")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Weekly Deaths
```

```
sns.lineplot(ax = axes[1], data = df_italy_weekly_1, x = df_italy_weekly_1.
  ↪index, y = df_italy_weekly_1["deaths"], color = 'red')
axes[1].set_title("Weekly COVID-19 Deaths in Italy")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

# Plot Weekly Recoveries
sns.lineplot(ax = axes[2], data = df_italy_weekly_1, x = df_italy_weekly_1.
  ↪index, y = df_italy_weekly_1["recovered"], color = 'green')
axes[2].set_title("Weekly COVID-19 Recoveries in Italy")
axes[2].set_xlabel("Date")
axes[2].set_ylabel("Recoveries")
axes[2].grid(True)

plt.tight_layout()
plt.show()
```



[30]:
```
# Create a figure with two subplots
fig, axes = plt.subplots(2, 1, figsize = (12, 8))

# Plot Weekly Cases
sns.lineplot(ax = axes[0], data = df_italy_weekly_2, x = df_italy_weekly_2.
 ↪index, y = df_italy_weekly_2["cases"], color = 'blue')
axes[0].set_title("Weekly COVID-19 Cases in Italy")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Weekly Deaths
sns.lineplot(ax = axes[1], data = df_italy_weekly_2, x = df_italy_weekly_2.
 ↪index, y = df_italy_weekly_2["deaths"], color = 'red')
axes[1].set_title("Weekly COVID-19 Deaths in Italy")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



We can observe that the overall trends remain consistent with those seen in the daily plots. In particular:

- There are higher mortality rates from January 2020 to August 2021. During this period, the ratio of deaths to cases was significantly higher, especially in the early waves of the pandemic. As mentioned earlier, this is likely due to limited testing, overwhelmed healthcare systems, and the absence of vaccines in the initial phase.
- After August 2021 there is a substantial increase in infections, yet the number of deaths does not rise proportionally. This is probably due to vaccination, improved treatments, and the emergence of potentially less lethal variants of the virus.
- The number of reported cases consistently declines during the summer months, probably due to increased outdoor activities and to seasonal variations in the transmission of the virus.

These observations highlight the essential role of medical advancements and behavioral adaptations in shaping the progression of the pandemic over time.

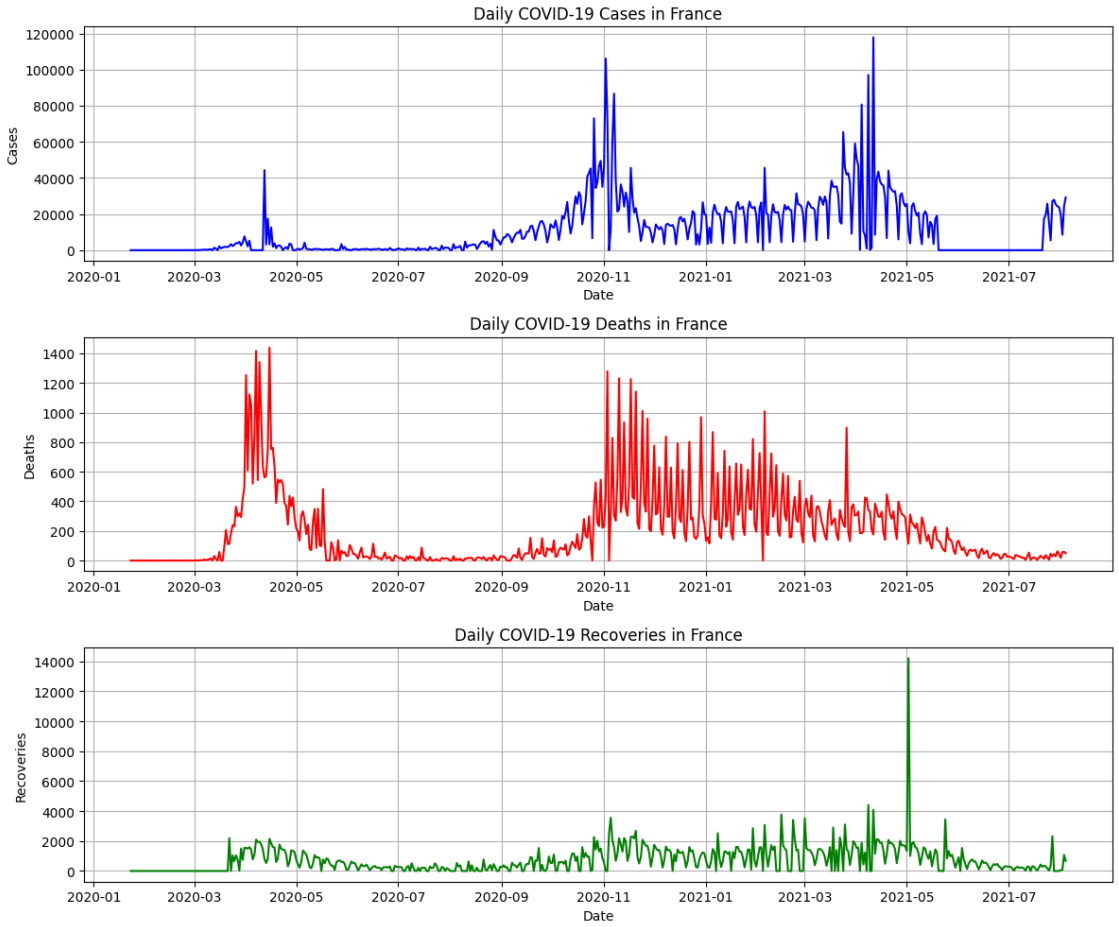
4.2 France

We repeat the exact same procedure for France.

We begin by analyzing the DataFrame that includes all variables up to 04-08-2021. We examine first the daily trends of the three variables, which are stored in the DataFrame `df_france_1_daily`. As before, we plot each variable separately using Seaborn.

[31]:
```
# Create a figure with three subplots
fig, axes = plt.subplots(3, 1, figsize = (12, 10))

# Plot Daily Cases
sns.lineplot(ax = axes[0], data = df_france_1_daily, x = df_france_1_daily.
 ↪index, y = df_france_1_daily["cases"], color = 'blue')
axes[0].set_title("Daily COVID-19 Cases in France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Daily Deaths
sns.lineplot(ax = axes[1], data = df_france_1_daily, x = df_france_1_daily.
 ↪index, y = df_france_1_daily["deaths"], color = 'red')
axes[1].set_title("Daily COVID-19 Deaths in France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

# Plot Daily Recoveries
sns.lineplot(ax = axes[2], data = df_france_1_daily, x = df_france_1_daily.
 ↪index, y = df_france_1_daily["recovered"], color = 'green')
axes[2].set_title("Daily COVID-19 Recoveries in France")
axes[2].set_xlabel("Date")
axes[2].set_ylabel("Recoveries")
axes[2].grid(True)
```

```
plt.tight_layout()
plt.show()
```

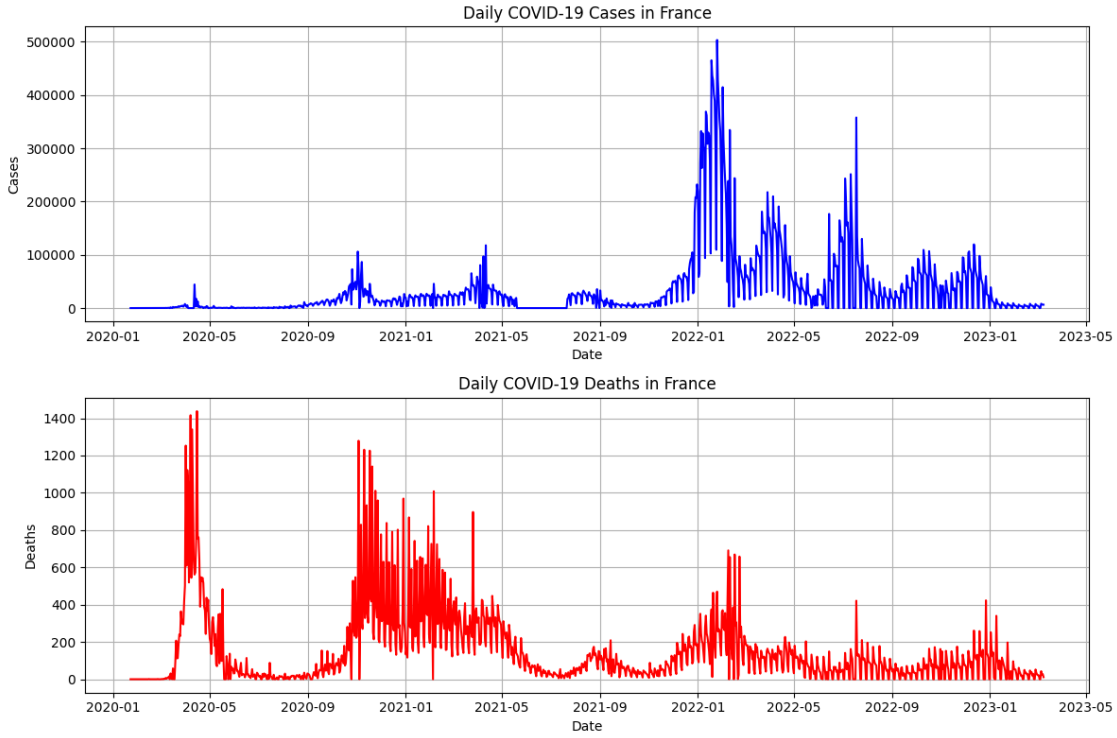


We can observe that:

- The above plots are challenging to interpret due to the high level of fluctuation. However, even in this case — particularly for the variables `cases` and `deaths` — we can distinguish three main pandemic waves, each marked by an increased number of infections and deaths. These roughly correspond to the periods March 2020 – May 2020, October 2020 – December 2020, and March 2021 – May 2021.
- As for the variable `recovered`, there is a value in May 2021 that is significantly higher than all other values. This is likely an outlier, but its extreme magnitude skews the scale of the graph, making it difficult to analyse the overall trend properly.
- As observed for Italy, the ratio of deaths to cases was significantly higher during the first wave compared to the subsequent ones. This is likely due to improved testing, medical advancements, the implementation of restrictions, and changes in public behavior. Indeed, the early phase of the pandemic was characterized by limited knowledge of the virus, overwhelmed healthcare systems, and fewer available treatments, all contributing to a higher mortality rate.

Let us now analyse the DataFrame containing only the variables `cases` and `deaths` for the whole time period (`df_france_2_daily`).

[32]:
```
# Create a figure with two subplots
fig, axes = plt.subplots(2, 1, figsize = (12, 8))

# Plot Daily Cases
sns.lineplot(ax = axes[0], data = df_france_2_daily, x = df_france_2_daily.
 ↪index, y = df_france_2_daily["cases"], color = 'blue')
axes[0].set_title("Daily COVID-19 Cases in France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Daily Deaths
sns.lineplot(ax = axes[1], data = df_france_2_daily, x = df_france_2_daily.
 ↪index, y = df_france_2_daily["deaths"], color = 'red')
axes[1].set_title("Daily COVID-19 Deaths in France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



The line plots above cover the entire period from January 2020 to March 2023. As with the previous plots, these are also challenging to interpret due to the high level of fluctuation. Nevertheless, it is clear that additional waves occurred after 04-08-2021, with daily cases rising sharply compared to the first half of the graph. Moreover, as observed for Italy, despite this surge in infections, the number of daily deaths remained lower than in previous waves, likely due to vaccination, improved treatments, and changes in the virus's characteristics.

Additionally, we note the following:

- There is a period between June 2021 and August 2021 where the number of daily cases is consistently zero. This is a consequence of how we initially handled the issue of negative values in the `df_france_daily` DataFrame. Specifically, we recall that a negative value appeared on 20-05-2021, and the error propagated forward. While having a continuous line of zeros is not ideal, it does not significantly impact the overall trend of the variable over the entire time period, so we can disregard it. To obtain a smoother trend, one could consider applying a linear interpolation between the values at the beginning and end of this period.
- Once again, we observe an unusual pattern in the final part of both plots, where both variables repeatedly drop to zero. To investigate this anomaly further, we print the last three weeks of data.

[33]:
```
df_france_2_daily.tail(21)
```

[33]:

	cases	deaths
2023-02-17	3589.0	28.0
2023-02-18	0.0	0.0
2023-02-19	0.0	0.0
2023-02-20	8133.0	47.0
2023-02-21	5774.0	32.0
2023-02-22	4602.0	38.0
2023-02-23	4489.0	19.0
2023-02-24	3917.0	29.0
2023-02-25	0.0	0.0
2023-02-26	0.0	0.0
2023-02-27	7626.0	52.0
2023-02-28	5651.0	34.0
2023-03-01	4525.0	25.0
2023-03-02	4196.0	21.0
2023-03-03	3194.0	21.0
2023-03-04	0.0	0.0
2023-03-05	0.0	0.0
2023-03-06	8146.0	43.0
2023-03-07	7063.0	24.0
2023-03-08	6111.0	27.0
2023-03-09	6308.0	11.0

As we can notice, this irregular pattern is due to the data not being updated during weekends.

As before, we now aggregate the data on a weekly basis to smooth out the daily fluctuations and

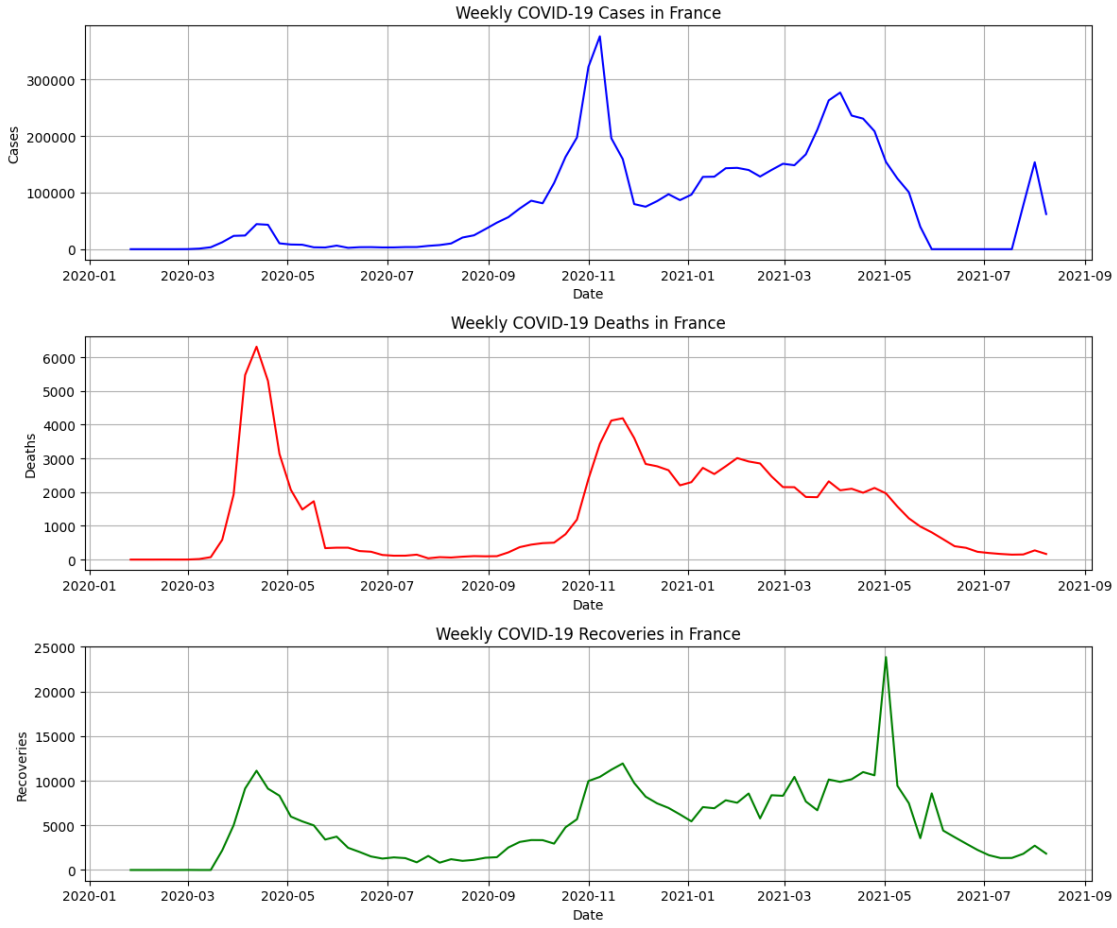
highlight long-term trends. We create the following new DataFrames:

- `df_france_weekly_1`: Aggregating data from `df_france_1_daily`, which includes cases, deaths, and recoveries up to August 2021.
- `df_france_weekly_2`: Aggregating data from `df_france_2_daily`, which contains cases and deaths up to March 2023.

[34]:
```
# Perform weekly aggregation by summing the daily values within each week
df_france_weekly_1 = df_france_1_daily.resample('W').sum()
df_france_weekly_2 = df_france_2_daily.resample('W').sum()
```

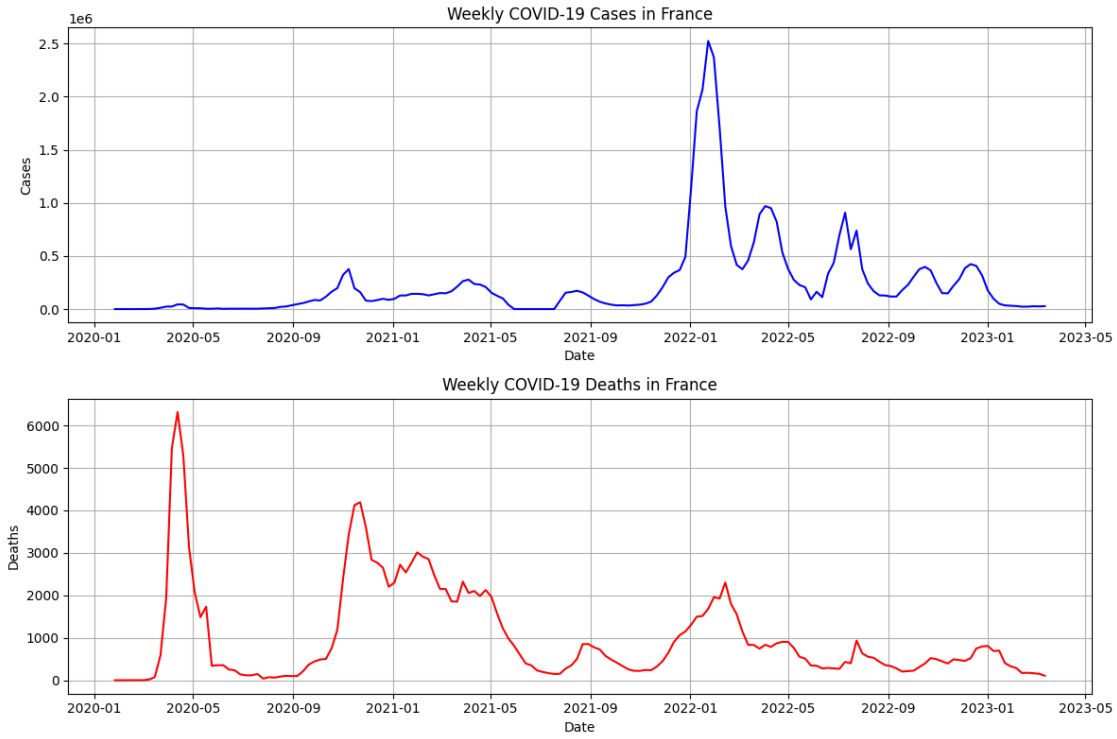
Below are the line plots showing the weekly trends for both datasets.

[35]:
```
# Create a figure with three subplots
fig, axes = plt.subplots(3, 1, figsize = (12, 10))

# Plot Weekly Cases
sns.lineplot(ax = axes[0], data = df_france_weekly_1, x = df_france_weekly_1.
 ↪index, y = df_france_weekly_1["cases"], color = 'blue')
axes[0].set_title("Weekly COVID-19 Cases in France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Weekly Deaths
sns.lineplot(ax = axes[1], data = df_france_weekly_1, x = df_france_weekly_1.
 ↪index, y = df_france_weekly_1["deaths"], color = 'red')
axes[1].set_title("Weekly COVID-19 Deaths in France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)

# Plot Weekly Recoveries
sns.lineplot(ax = axes[2], data = df_france_weekly_1, x = df_france_weekly_1.
 ↪index, y = df_france_weekly_1["recovered"], color = 'green')
axes[2].set_title("Weekly COVID-19 Recoveries in France")
axes[2].set_xlabel("Date")
axes[2].set_ylabel("Recoveries")
axes[2].grid(True)

plt.tight_layout()
plt.show()
```



[36]:
```
# Create a figure with two subplots
fig, axes = plt.subplots(2, 1, figsize = (12, 8))

# Plot Weekly Cases
sns.lineplot(ax = axes[0], data = df_france_weekly_2, x = df_france_weekly_2.
 ↪index, y = df_france_weekly_2["cases"], color = 'blue')
axes[0].set_title("Weekly COVID-19 Cases in France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].grid(True)

# Plot Weekly Deaths
sns.lineplot(ax = axes[1], data = df_france_weekly_2, x = df_france_weekly_2.
 ↪index, y = df_france_weekly_2["deaths"], color = 'red')
axes[1].set_title("Weekly COVID-19 Deaths in France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].grid(True)
```

```
plt.tight_layout()
plt.show()
```



We can observe that aggregating the data on a weekly basis makes the trends much easier to visualize compared to the daily plots. The overall patterns remain consistent with those seen in the daily data, they are simply clearer and more interpretable. In particular, we can now identify with more clarity the three waves from January 2020 to August 2021, as well as the subsequent ones. Notably, the plot for the variable `recovered` is significantly more readable, as weekly aggregation reduces the skewness caused by the May 2021 outlier, allowing us to better distinguish the three waves observed in the daily plots.

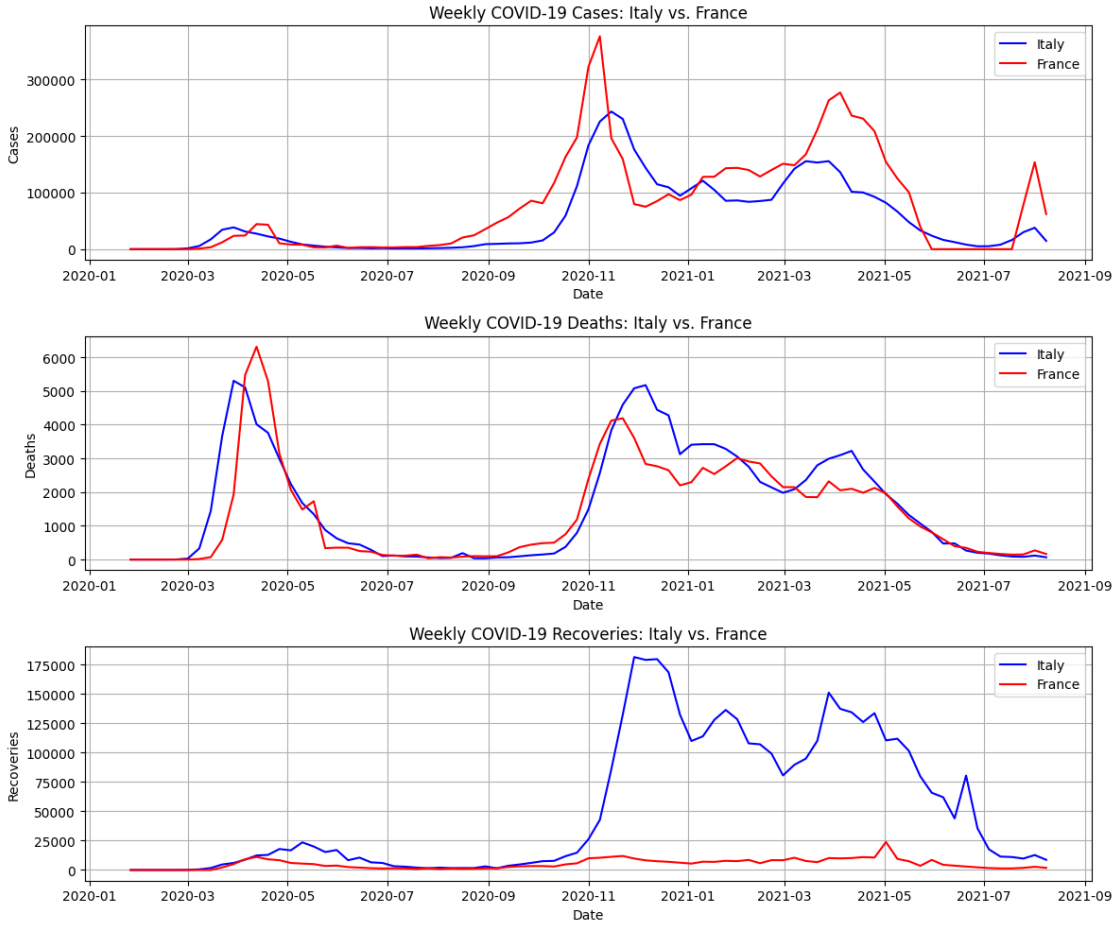
5 Comparative visualizations

To compare the progression of COVID-19 metrics in Italy and France, we use weekly trends, as they are smoother and easier to visualize and compare than daily trends.

We begin by analysing the DataFrames containing the weekly trends of all variables up to 04-08-2021 (`df_italy_weekly_1`, `df_france_weekly_1`). For each variable, we utilize both line plots for precise trend comparisons and area plots to highlight the overall magnitude.

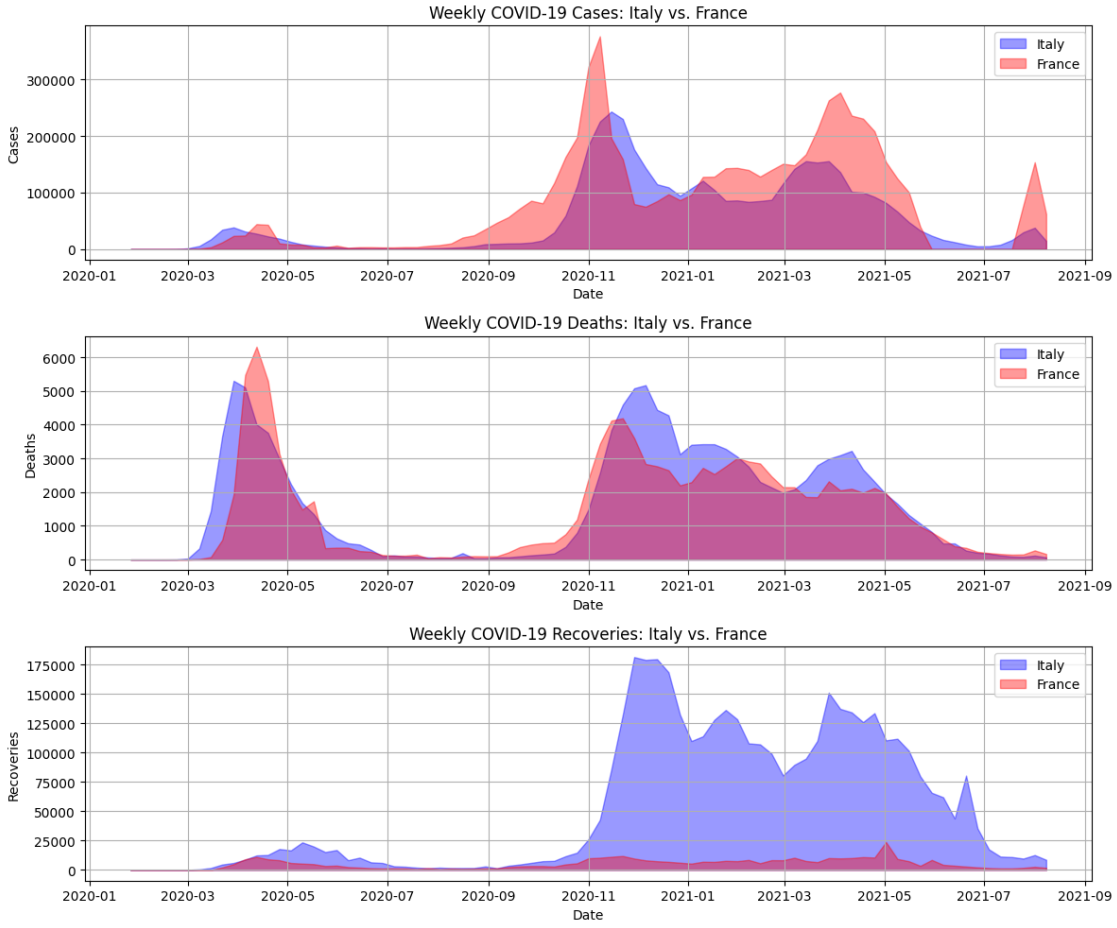
[37]:
```
# Create a figure with three subplots
fig, axes = plt.subplots(3, 1, figsize = (12, 10))
```

```
# Define colors
colors = {"Italy": "blue", "France": "red"}

# Plot Weekly Cases (Line Plot)
sns.lineplot(ax = axes[0], x = df_italy_weekly_1.index, y =␣
 ↪df_italy_weekly_1["cases"], color = colors["Italy"], label = "Italy")
sns.lineplot(ax = axes[0], x = df_france_weekly_1.index, y =␣
 ↪df_france_weekly_1["cases"], color = colors["France"], label = "France")
axes[0].set_title("Weekly COVID-19 Cases: Italy vs. France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].legend()
axes[0].grid(True)

# Plot Weekly Deaths (Line Plot)
sns.lineplot(ax = axes[1], x = df_italy_weekly_1.index, y =␣
 ↪df_italy_weekly_1["deaths"], color = colors["Italy"], label = "Italy")
sns.lineplot(ax = axes[1], x = df_france_weekly_1.index, y =␣
 ↪df_france_weekly_1["deaths"], color = colors["France"], label = "France")
axes[1].set_title("Weekly COVID-19 Deaths: Italy vs. France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].legend()
axes[1].grid(True)

# Plot Weekly Recoveries (Line Plot)
sns.lineplot(ax = axes[2], x = df_italy_weekly_1.index, y =␣
 ↪df_italy_weekly_1["recovered"], color = colors["Italy"], label = "Italy")
sns.lineplot(ax = axes[2], x = df_france_weekly_1.index, y =␣
 ↪df_france_weekly_1["recovered"], color = colors["France"], label = "France")
axes[2].set_title("Weekly COVID-19 Recoveries: Italy vs. France")
axes[2].set_xlabel("Date")
axes[2].set_ylabel("Recoveries")
axes[2].legend()
axes[2].grid(True)

plt.tight_layout()
plt.show()
```



[38]:
```
# Create a figure with three subplots
fig, axes = plt.subplots(3, 1, figsize = (12, 10))

# Plot Weekly Cases (Area Plot)
axes[0].fill_between(df_italy_weekly_1.index, df_italy_weekly_1["cases"], color␣
 ↪= colors["Italy"], alpha = 0.4, label = "Italy")
axes[0].fill_between(df_france_weekly_1.index, df_france_weekly_1["cases"],␣
 ↪color = colors["France"], alpha = 0.4, label = "France")
axes[0].set_title("Weekly COVID-19 Cases: Italy vs. France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].legend()
axes[0].grid(True)

# Plot Weekly Deaths (Area Plot)
axes[1].fill_between(df_italy_weekly_1.index, df_italy_weekly_1["deaths"],␣
 ↪color = colors["Italy"], alpha = 0.4, label = "Italy")
```

```
axes[1].fill_between(df_france_weekly_1.index, df_france_weekly_1["deaths"],␣
 ↪color = colors["France"], alpha = 0.4, label = "France")
axes[1].set_title("Weekly COVID-19 Deaths: Italy vs. France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].legend()
axes[1].grid(True)

# Plot Weekly Recoveries (Area Plot)
axes[2].fill_between(df_italy_weekly_1.index, df_italy_weekly_1["recovered"],␣
 ↪color = colors["Italy"], alpha = 0.4, label = "Italy")
axes[2].fill_between(df_france_weekly_1.index, df_france_weekly_1["recovered"],␣
 ↪color = colors["France"], alpha = 0.4, label = "France")
axes[2].set_title("Weekly COVID-19 Recoveries: Italy vs. France")
axes[2].set_xlabel("Date")
axes[2].set_ylabel("Recoveries")
axes[2].legend()
axes[2].grid(True)

plt.tight_layout()
plt.show()
```



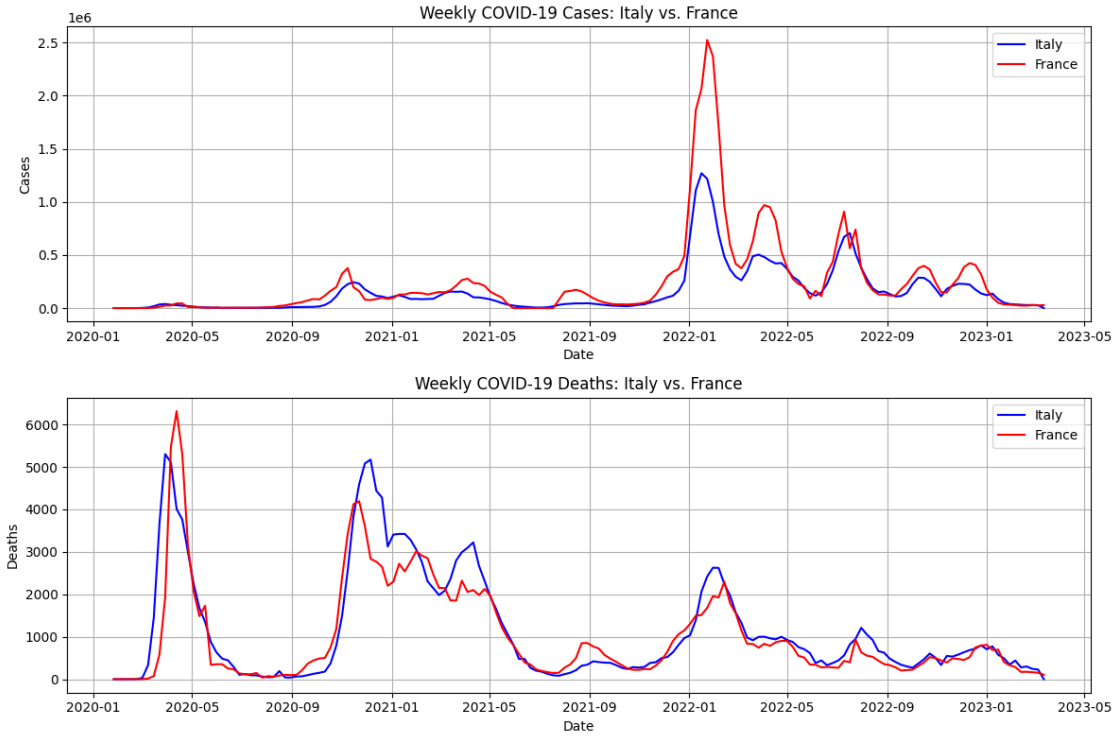
The above plots show that:

- For the variable `cases`, Italy and France exhibit similar trends. In general, the weekly number of cases in Italy appears to be lower, with France consistently reaching higher peaks than Italy, particularly during the periods October - November 2020 and April - May 2021.
- For the variable `deaths`, Italy and France again show similar trends. However, the number of deaths is often higher in Italy. While France reaches a higher peak in April 2020, in the second half of the plot, Italy's peaks tend to be higher.
- For the variable `recovered`, the trends are very similar up until November 2020. After that, while France's weekly recovery numbers remain stable, Italy experiences a significant increase, leading to very different trends and volumes in the second half of the plot.

Moreover, we can also notice that the weekly trends in the number of cases and deaths are very similar, which makes perfect sense since a higher number of cases generally leads to a higher number of deaths, with a certain delay.

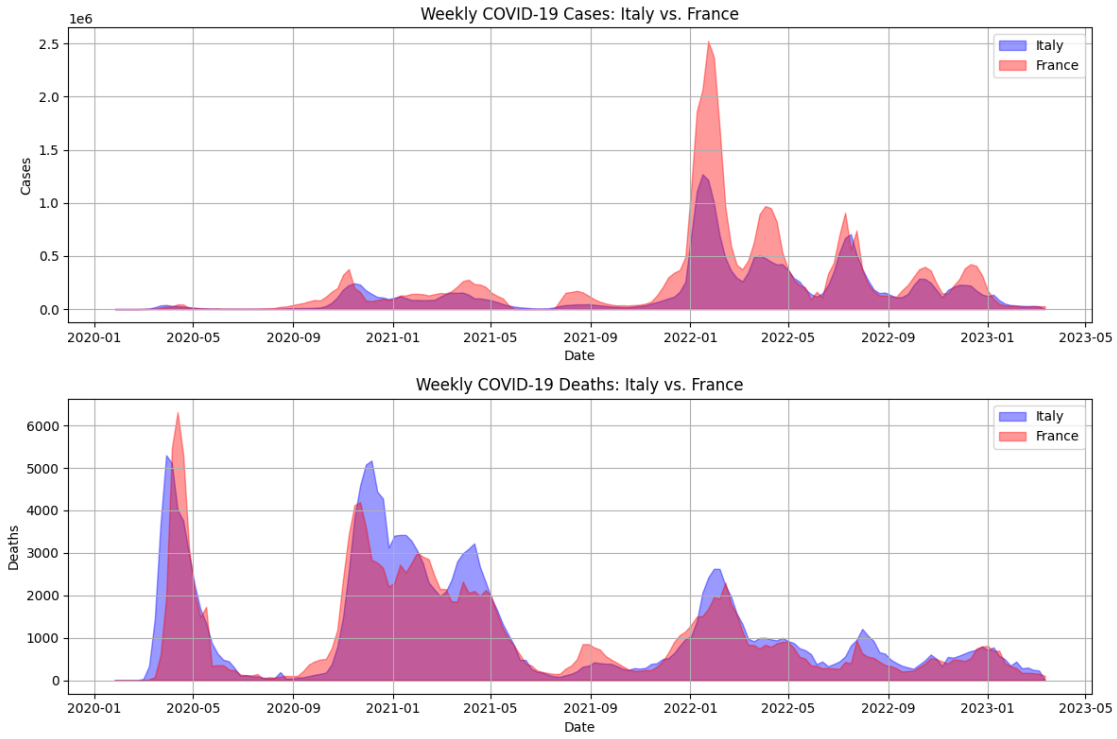
We now analyse the DataFrames containing the weekly trends of only the variables `cases` and `deaths` for the entire time period (`df_italy_weekly_2`, `df_france_weekly_2`).

[39]:
```
# Create a figure with two subplots
fig, axes = plt.subplots(2, 1, figsize = (12, 8))

# Plot Weekly Cases (Line Plot)
sns.lineplot(ax = axes[0], x = df_italy_weekly_2.index, y =␣
 ↪df_italy_weekly_2["cases"], color = colors["Italy"], label = "Italy")
sns.lineplot(ax = axes[0], x = df_france_weekly_2.index, y =␣
 ↪df_france_weekly_2["cases"], color = colors["France"], label = "France")
axes[0].set_title("Weekly COVID-19 Cases: Italy vs. France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].legend()
axes[0].grid(True)

# Plot Weekly Deaths (Line Plot)
sns.lineplot(ax = axes[1], x = df_italy_weekly_2.index, y =␣
 ↪df_italy_weekly_2["deaths"], color = colors["Italy"], label = "Italy")
sns.lineplot(ax = axes[1], x = df_france_weekly_2.index, y =␣
 ↪df_france_weekly_2["deaths"], color = colors["France"], label = "France")
axes[1].set_title("Weekly COVID-19 Deaths: Italy vs. France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].legend()
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



[40]:
```
# Create a figure with two subplots
fig, axes = plt.subplots(2, 1, figsize = (12, 8))

# Plot Weekly Cases (Area Plot)
axes[0].fill_between(df_italy_weekly_2.index, df_italy_weekly_2["cases"], color␣
 ↪= colors["Italy"], alpha = 0.4, label = "Italy")
axes[0].fill_between(df_france_weekly_2.index, df_france_weekly_2["cases"],␣
 ↪color = colors["France"], alpha = 0.4, label = "France")
axes[0].set_title("Weekly COVID-19 Cases: Italy vs. France")
axes[0].set_xlabel("Date")
axes[0].set_ylabel("Cases")
axes[0].legend()
axes[0].grid(True)

# Plot Weekly Deaths (Area Plot)
axes[1].fill_between(df_italy_weekly_2.index, df_italy_weekly_2["deaths"],␣
 ↪color = colors["Italy"], alpha = 0.4, label = "Italy")
axes[1].fill_between(df_france_weekly_2.index, df_france_weekly_2["deaths"],␣
 ↪color = colors["France"], alpha = 0.4, label = "France")
axes[1].set_title("Weekly COVID-19 Deaths: Italy vs. France")
axes[1].set_xlabel("Date")
axes[1].set_ylabel("Deaths")
axes[1].legend()
```

```
axes[1].grid(True)

plt.tight_layout()
plt.show()
```



As we can see from the above plots, the trends after 04-08-2021 remain similar in Italy and France for both variables. As before, the weekly number of cases in Italy is often lower than in France, with France consistently reaching higher peaks, particularly in January 2022 and April 2022. Similarly, the weekly number of deaths is generally higher in Italy; except for April 2020 and September 2021, where France experiences higher peaks, Italy's peaks tend to be slightly higher throughout the rest of the period.

6 Conclusions

Our analysis of COVID-19 progression in Italy and France highlights several key similarities and differences in the trends of cases, deaths, and recoveries. Both countries experienced similar pandemic waves, but with some differences in magnitude:

- France consistently reported higher peaks in weekly cases compared to Italy, suggesting either higher transmission rates or differences in testing strategies between the two countries.
- While both countries exhibited similar mortality trends, Italy generally recorded higher peaks in deaths, except during April 2020 and September 2021, where France reached higher peaks. This could indicate differences in the healthcare system or can be due to demographic factors.

- Until November 2020, recovery trends were similar in both countries. However, after this point, Italy saw a substantial rise in reported recoveries, diverging from the relatively stable trend observed in France. This discrepancy may be due to differences in how recoveries were recorded or reported.
- In both countries, cases tended to decline during the summer months, likely due to behavioral changes, increased outdoor activities, and possible seasonal variations in the virus transmission. Additionally, after August 2021, despite a sharp rise in infections, the number of deaths remained lower than in previous waves, likely due to the impact of vaccination, improved treatments, and changes in the virus's characteristics.

Overall, while Italy and France followed similar trajectories, variations in case numbers, mortality peaks, and recovery trends highlight country-specific differences in the pandemic response.